Mathematics of Neural Networks

Volume III: Graph, Geometric and Multimodal Learning

Structure, symmetry, and cross-modal mathematics

Every derivation step by step

Every symbol defined

Explicit numerical examples

Geometric intuitions

Silvano Salvador

Contents

25.1 Graphs: Formal Definitions 2

25.2 The Graph Laplacian 3

25.3 Graph Fourier Transform 4

25.4 Spectral Gap and Mixing Time 6

25.5 Random Walk Graph Kernels 7

26.1 Spectral Convolution 9

26.2 ChebNet: Polynomial Approximation 10

26.3 Graph Convolutional Network (GCN) 11

26.4 Graph Isomorphism Network (GIN) 12

27.1 Graph Attention Network (GAT) 14

27.2 GraphSAGE: Inductive Learning on Large Graphs 15

27.3 GraphSAGE: Sampling and Aggregation 16

27.4 Heterogeneous Graphs and HAN 17

28.1 Groups and Symmetries 18

28.2 Invariance and Equivariance 19

28.3 Group Convolution (G-CNN) 20

28.4 Representations of Groups and Schur's Lemma 21

28.5 The Wigner D-Matrix and Spherical Harmonics 22

29.1 The SE(3) Equivariance Problem 23

29.2 EGNN Propagation Rules 25

30.1 The Monge-Kantorovich Problem 27

30.2 Discrete Optimal Transport 28

30.3 Sinkhorn: Entropic Regularisation 29

30.4 Wasserstein Barycenter 30

30.5 Gromov-Wasserstein for Cross-Domain Comparison 31

30.6 Sliced Wasserstein Distance 33

31.1 Contrastive Learning and InfoNCE 34

31.2 Cross-Attention for Vision-Language Fusion 35

31.3 CLIP: Contrastive Language-Image Pre-Training 36

31.4 Flamingo: Few-Shot Vision-Language Models 37

32.1 Volume Rendering 38

32.2 Positional Encoding for High-Frequency Detail 39

32.3 SIREN: Sinusoidal Activations for Implicit Representations 40

33.1 Multiple Sequence Alignment and Evolutionary Attention 41

33.2 The Invariant Point Attention (IPA) Module 43

33.3 Backbone Frame Update 44

33.4 Template Pair Stack and Structure Module 45

34.1 Smooth Manifolds 46

34.2 Geodesics and Exponential Map 47

34.3 Persistent Homology 48

34.4 Riemannian Gradient Descent 49

34.5 Topological Data Analysis and Persistence 50

35.1 The Five G's of Geometric Deep Learning 51

35.2 Message Passing Neural Networks (MPNN) 53

35.3 Expressive Power: The Weisfeiler-Lehman Test 54

35.4 Graph Transformers 55

35.5 Summary of Volume III 56

36.1 Transformer Architecture: Full Mathematical Derivation 58

36.2 Rotary Positional Embedding (RoPE) and ALiBi 60

36.3 Flash Attention: Hardware-Aware Algorithm 61

36.4 Sparse and Linear Attention Variants 62

37.1 Score-Based Diffusion Models: Mathematical Framework 63

37.2 Flow Matching and Rectified Flows 65

37.3 Classifier-Free Guidance and DDIM Sampling 66

37.4 Latent Diffusion and Stable Diffusion 67

38.1 Neural Scaling Laws 68

38.2 Hyperbolic Neural Networks 70

38.3 Higher-Order GNNs and Graph Transformers 71

38.4 Position Encoding on Graphs: Laplacian Eigenvectors 72

38.5 Summary Table: GNN Architectures and Expressiveness 73

38.6 Connections to Classical Computer Science 74

38.7 Benchmark Datasets and Evaluation Protocol 75

38.8 Summary and Open Problems in Geometric Deep Learning 76

39.1 Graph Generation: VGAE, GraphRNN, and DiGress 78

39.2 Neural Combinatorial Optimisation on Graphs 79

39.3 Protein Structure Prediction: Beyond AlphaFold2 80

40.1 Multimodal Large Language Models: Architecture and Training 82

40.2 Geometric Deep Learning in Scientific Discovery 83

41.1 The Spectral-Spatial Duality in Graph Learning 85

41.2 Equivariance as a Unifying Principle 85

41.3 Mathematical Maturity and Prerequisites for Volume IV 86

Dimostrazioni Differite 87

Ricettario: Operazioni su Grafi e Geometria 89

Complessità: GNN, Attenzione e Trasporto Ottimale 90

Glossario dei Simboli — Volume III 91

Bibliografia 92

PART 25

Graph Theory and the Laplacian

We define graphs formally, derive the graph Laplacian and its spectral decomposition, and show how the spectrum encodes connectivity.

25.1 Graphs: Formal Definitions

Definition: Graph

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of:

- $\mathcal{V} = \{1, \dots, N\}$: set of N *nodes* (vertices).
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$: set of M *edges*.

An *undirected* graph has $(i,j) \in \mathcal{E} \Leftrightarrow (j,i) \in \mathcal{E}$. Node features: $\mathbf{X} \in \mathbb{R}^{N \times d}$, row i is the d-dimensional feature of node i.

Rappresentazioni matriciali. La *matrice di adiacenza* $\mathbf{A} \in \{0,1\}^{N \times N}$ ha $A_{ij} = 1$ sse $(i,j) \in \mathcal{E}$. Per grafi pesati $A_{ij} \in \mathbb{R}_{\geq 0}$. Il *grado* del nodo i è $d_i = \sum_j A_{ij}$; la *matrice dei gradi* $\mathbf{D} = \mathrm{diag}(d_1, \dots, d_N)$.

Proprietà strutturali.

- *Densità*: $\rho = 2M/N(N-1)$. Grafi reali tipicamente sparsi: $\rho \ll 1$.
- *Diametro*: lunghezza del percorso più lungo tra i percorsi minimi.
- *Coefficiente di clustering*: $C_i = \frac{2|\{(j,k) \in \mathcal{E} : j,k \in \mathcal{N}(i)\}|}{d_i(d_i-1)}$, misura quanto i vicini di i siano connessi tra loro.
- *Componenti connesse*: partizione di $\mathcal{V}$ in sottografi massimali connessi.

Grafi etichettati e multirelazionali. In molte applicazioni (knowledge graph, reti molecolari) gli archi hanno tipo $r \in \mathcal{R}$, dando luogo a *grafi eterogenei* con matrici di adiacenza separate $\mathbf{A}^{(r)}$ per ogni relazione.

Example — Molecole come grafi

In cheminformatica: nodi = atomi (feature: numero atomico, elettronegatività), archi = legami (feature: ordine di legame, lunghezza). Una molecola come l'etanolo C_2H_5OH diventa un grafo con 9 nodi. Questa rappresentazione è invariante per permutazione degli indici, requisito fondamentale per le reti equivarianti.

Definition: Weighted Graph and Incidence Matrix

A weighted graph assigns a weight $w_{ij} \in \mathbb{R}_{>0}$ to each edge $(i,j) \in \mathcal{E}$. The weighted adjacency matrix has $A_{ij} = w_{ij}$.
The *incidence matrix* $\mathbf{B} \in \{-1, 0, 1\}^{M \times N}$ (oriented) has $B_{ej} = +1$ if edge e originates at node j and $B_{ej} = -1$ if it terminates at j. One can verify that $\mathbf{L} = \mathbf{B}^\top \mathbf{B}$ for the combinatorial Laplacian.

Adjacency matrix. $\mathbf{A} \in \{0,1\}^{N \times N}$ with $A_{ij} = 1$ iff $(i,j) \in \mathcal{E}$.

Degree matrix. $\mathbf{D} = \mathrm{diag}(d_1, \dots, d_N)$ where $d_i = \sum_j A_{ij}$ is the degree of node i.

Walks, paths, and cycles. A *walk* of length k from i to j is a sequence of nodes $i = v_0, v_1, \dots, v_k = j$ with each consecutive pair connected by an edge. The (i, j) entry of $\mathbf{A}^k$ equals the number of walks of length k from i to j. A *path* is a walk with no repeated nodes; a *cycle* is a closed walk with no repeated nodes. These combinatorial quantities encode connectivity and are directly linked to spectral properties.

Example — Counting triangles via the spectrum

The number of triangles in a graph is $\frac{1}{6}\mathrm{tr}(\mathbf{A}^3) = \frac{1}{6}\sum_k \lambda_k^3$, where $\{\lambda_k\}$ are the eigenvalues of $\mathbf{A}$. This shows that global structural properties are encoded in the spectrum. For the complete graph K_N: $\lambda_1 = N-1$ (multiplicity 1), $\lambda_k = -1$ (multiplicity $N-1$), giving $\frac{1}{6}[(N-1)^3 + (N-1)(-1)^3] = \binom{N}{3}$ triangles as expected.

25.2 The Graph Laplacian

Laplaciano combinatorio

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \qquad L_{ij} = \begin{cases} d_i & i = j \\ -1 & (i,j) \in \mathcal{E} \\ 0 & \text{altrimenti} \end{cases}$$

Proprietà fondamentali di L.

- Simmetrica e semidefinita positiva (SDP): $\mathbf{x}^\top \mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}}(x_i - x_j)^2 \geq 0$.
- Gli autovalori reali $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$ si chiamano *spettro del grafo*.
- Molteplicità di $\lambda = 0$ = numero di componenti connesse.
- Il vettore costante $\mathbf{1}$ è sempre autovettore per $\lambda = 0$.

Laplaciano normalizzato simmetrico.

$$\tilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$$

Autovalori in $[0, 2]$. Vettori propri ortonormali $\mathbf{U} \in \mathbb{R}^{N\times N}$, $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$.

Interpretazione variazionale. Il secondo autovettore $\mathbf{u}_2$ (vettore di Fiedler) minimizza:

$$\min_{\mathbf{x}\perp\mathbf{1},\|\mathbf{x}\|=1} \mathbf{x}^\top\mathbf{L}\mathbf{x} = \min_{\mathbf{x}\perp\mathbf{1},\|\mathbf{x}\|=1} \sum_{(i,j)\in\mathcal{E}} (x_i - x_j)^2$$

Questa quantità (λ_2, connettività algebrica) misura quanto è difficile disconnettere il grafo. È la base del *spectral clustering* e della *graph partitioning*.

Derivazione dell'equazione del calore su grafi. Analoga all'equazione del calore continua $\partial_t u = \Delta u$, su grafi:

$$\frac{d\mathbf{x}}{dt} = -\mathbf{L}\mathbf{x}$$

Soluzione: $\mathbf{x}(t) = e^{-t\mathbf{L}}\mathbf{x}(0) = \mathbf{U}e^{-t\Box}\mathbf{U}^\top\mathbf{x}(0)$. Il segnale si diffonde smoothly sul grafo; le componenti ad alta frequenza (grandi λ_k) decadono più rapidamente.

Definition: Graph Laplacian

The (combinatorial) Laplacian: $\mathbf{L} = \mathbf{D} - \mathbf{A}$
The normalised Laplacian: $\hat{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$

Properties of L:

- $\mathbf{L}$ is symmetric and positive semi-definite: $\mathbf{f}^\top \mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}}(f_i - f_j)^2 \geq 0$.
- Eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$.
- Multiplicity of eigenvalue 0 equals the number of connected components.
- The eigenvector for $\lambda_1 = 0$ is $\mathbf{1}/\sqrt{N}$ (constant signal).

Derivation of $\mathbf{f}^\top\mathbf{L}\mathbf{f}$:

$$\mathbf{f}^\top \mathbf{L}\mathbf{f} = \mathbf{f}^\top \mathbf{D}\mathbf{f} - \mathbf{f}^\top \mathbf{A}\mathbf{f} = \sum_i d_i f_i^2 - \sum_{i,j} A_{ij} f_i f_j$$
$$= \sum_{(i,j)\in\mathcal{E}} (f_i^2 + f_j^2 - 2f_i f_j) = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} (f_i - f_j)^2$$

This is the *Dirichlet energy*: it measures how much the signal $\mathbf{f}$ varies across edges.

Example — Laplacian of a path graph P_4

Nodes $\{1,2,3,4\}$, edges $\{(1,2),(2,3),(3,4)\}$:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{D} = \mathrm{diag}(1,2,2,1), \quad \mathbf{L} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

Eigenvalues: $\lambda_1 = 0$ (connected, one component), $\lambda_2 \approx 0.586$, $\lambda_3 \approx 2$, $\lambda_4 \approx 3.414$.
Dirichlet energy of $\mathbf{f} = (1,0,0,0)^\top$: $(1-0)^2 = 1$ (signal jumps at edge $(1,2)$).

The random walk Laplacian. A third variant is the random walk Laplacian $\mathbf{L}_{\text{rw}} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$. Unlike $\hat{\mathbf{L}}$, it is not symmetric, but its eigenvalues are the same as those of $\hat{\mathbf{L}}$ (the two are similar via $\mathbf{D}^{1/2}$). The operator $\mathbf{D}^{-1}\mathbf{A}$ is exactly the transition matrix of the simple random walk, so $\mathbf{L}_{\text{rw}}$ has a direct probabilistic interpretation.

Cheeger's inequality. The spectral gap λ_2 and the graph's edge conductance $h_G = \min_{S\subset\mathcal{V}} \frac{|\partial S|}{\min(\mathrm{vol}(S),\mathrm{vol}(\bar{S}))}$ satisfy:

$$\frac{h_G^2}{2} \leq \lambda_2 \leq 2h_G$$

This fundamental inequality links the algebraic connectivity λ_2 to the combinatorial bottleneck h_G. A graph with small h_G (easy to cut) has small λ_2; a well-connected expander graph has large λ_2.

25.3 Graph Fourier Transform

Sia $\mathbf{L} = \mathbf{U}\square\mathbf{U}^\top$ la decomposizione spettrale del Laplaciano normalizzato, con $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$ e $\square = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$.

Graph Fourier Transform (GFT)

$$\hat{\mathbf{x}} = \mathbf{U}^\top \mathbf{x} \qquad \text{(trasformata)}, \qquad \mathbf{x} = \mathbf{U}\hat{\mathbf{x}} \qquad \text{(antitrasformata)}$$

$\hat{x}_k = \langle \mathbf{x}, \mathbf{u}_k \rangle$ è il coefficiente di Fourier alla "frequenza" λ_k.

Analogia con la DFT classica. Per il grafo a cammino P_N (griglia 1D), gli autovettori del Laplaciano sono esattamente i modi $\sin(k\pi n/N)$, recuperando la trasformata coseno discreta (DCT). Il Laplaciano del grafo generalizza quindi il Laplaciano continuo $-\Delta$ agli spazi discreti irregolari.

Convoluzione spettrale. Su segnali euclidei: convoluzione $\equiv$ moltiplicazione in frequenza. Su grafi:

$$(\mathbf{x} *_{\mathcal{G}} \mathbf{y}) = \mathbf{U}(\hat{\mathbf{x}} \odot \hat{\mathbf{y}}) = \mathbf{U}\,\mathrm{diag}(\hat{\mathbf{y}})\,\mathbf{U}^\top \mathbf{x}$$

Il filtro g_θ è un vettore di N parametri nel dominio spettrale. Complessità: $O(N^2)$ per la moltiplicazione con $\mathbf{U}$, problematica per grafi grandi.

Limitazione della GFT. Gli autovettori del Laplaciano dipendono dalla struttura globale del grafo e non sono localizzati spazialmente: un segnale localizzato su un nodo ha GFT delocalizzata. Questo è il punto di partenza per le approssimazioni polinomiali (ChebNet, GCN).

The eigendecomposition $\mathbf{L} = \mathbf{U}\square\mathbf{U}^\top$ (with columns of $\mathbf{U}$ the eigenvectors) defines:

Graph Fourier Transform

Forward: $\hat{\mathbf{f}} = \mathbf{U}^\top \mathbf{f}$ (project signal onto eigenvectors)
Inverse: $\mathbf{f} = \mathbf{U}\hat{\mathbf{f}}$ (reconstruct from frequency components)

Analogy with classical Fourier. The eigenvectors of $\mathbf{L}$ play the role of complex exponentials $e^{2\pi ikt/N}$. Low-eigenvalue eigenvectors are smooth (slow variation), high-eigenvalue ones are oscillatory (rapid variation). Graph convolution in the spectral domain:

$$\mathbf{f} \star \mathbf{g} = \mathbf{U}(\hat{\mathbf{f}} \odot \hat{\mathbf{g}})$$

Total variation on graphs. The total variation of a signal $\mathbf{f}$ on a graph is:

$$\mathrm{TV}(\mathbf{f}) = \sum_{(i,j)\in\mathcal{E}} |f_i - f_j|$$

Compared to Dirichlet energy (which uses squared differences), TV penalises sharp discontinuities linearly. TV-based graph signal processing leads to robust filtering analogous to image denoising via total variation minimisation.

Parseval Frames and Tight Frames The graph Fourier transform requires the eigenvectors $\mathbf{U}$ to be an orthonormal basis, which exists only for undirected graphs with a symmetric Laplacian. For directed graphs, $\mathbf{L}$ is not symmetric and its eigenvectors may not be orthogonal. *Magnetic Laplacian* $\mathbf{L}^{(q)} = \mathbf{D} - \mathbf{A}^{(q)}$ with $\mathbf{A}^{(q)}_{ij} = A_{ij} e^{i2\pi q(A_{ij} - A_{ji})}$ is Hermitian positive semidefinite for any $q \in [0, \frac{1}{4}]$, providing a well-defined Fourier basis for directed graphs that encodes flow direction via complex phases.

Example — Graph Fourier Transform on a social network

For the Karate Club graph (34 nodes, 78 edges): the graph Laplacian $\mathbf{L}$ has 34 eigenvectors. The two lowest-frequency eigenvectors $\mathbf{u}_1, \mathbf{u}_2$ assign similar values to nodes in the same community (Zachary's two factions). A node signal $\mathbf{f}$ assigning $+1$ to faction A and -1 to faction B has its energy concentrated in $\hat{f}_2 = \mathbf{u}_2^\top \mathbf{f} \approx -4.7$ (the Fiedler component) — the community structure is a low-frequency graph signal. Applying a low-pass graph filter $h(\lambda) = \mathbf{1}[\lambda < \lambda_5]$ (zeroing the top 29 eigenvalues) reconstructs the community labels perfectly, illustrating that spectral graph filtering separates structural from noisy signals.

25.4 Spectral Gap and Mixing Time

Passeggiata casuale su grafi. La matrice di transizione è $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$. A partire dalla distribuzione iniziale $\square^{(0)}$, dopo t passi: $\square^{(t)} = \mathbf{P}^\top \square^{(0)}$ (o $\square^{(t)\top} = \square^{(0)\top}\mathbf{P}$). La distribuzione stazionaria (per grafi connessi non bipartiti) è $\square^* = \mathbf{d}/\mathrm{vol}(\mathcal{G})$ dove $\mathrm{vol}(\mathcal{G}) = \sum_i d_i$.

Spectral gap

$\gamma = 1 - \lambda_2(\mathbf{P})$ dove $\lambda_2(\mathbf{P})$ è il secondo autovalore più grande della matrice di transizione. Equivalentemente $\gamma = \lambda_2(\tilde{\mathbf{L}})$ (secondo autovalore più piccolo del Laplaciano normalizzato).

Mixing time. Il tempo di mescolamento è $t_{\text{mix}}(\varepsilon) = \min\{t : \max_x \|P^t(x,\cdot) - \pi\|_{\text{TV}} \le \varepsilon\}$. Il teorema spettrale dà:

$$t_{\text{mix}}(\varepsilon) = O\left(\frac{1}{\gamma}\log\frac{1}{\varepsilon\pi_{\min}}\right)$$

Spectral gap grande $\Rightarrow$ mescolamento rapido $\Rightarrow$ informazione si propaga efficientemente $\Rightarrow$ pochi layer GNN bastano.

Connessione con GNN depth. In un GNN con aggregazione media, dopo k layer il nodo i ha visto tutti i nodi nel *k-hop neighborhood* $\mathcal{N}^k(i)$. Su grafi con spectral gap piccolo (es. grafi espanditori vs. grafi a catena), il numero di layer necessario per propagare informazione globale differisce di un fattore $O(1/\gamma)$. I grafi con diametro grande richiedono GNN profondi o architetture con skip connections.

The spectral gap of the normalised Laplacian $\Delta = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is λ_2 (second-smallest eigenvalue). It controls how fast random walks on the graph mix:

$$t_{\text{mix}} = O\left(\frac{1}{\lambda_2}\log\frac{1}{\epsilon}\right)$$

Small λ_2 (near-disconnected graph) $\Rightarrow$ slow mixing $\Rightarrow$ hard for GNNs to propagate information.

Over-squashing. In GNNs, information from exponentially many nodes must be squashed into a fixed-size vector as depth increases. Bottleneck edges in the graph cause over-squashing: nodes far from hubs lose information. Spectral gap quantifies this.

Graph rewiring. Adding virtual edges (e.g. connecting nodes with small effective resistance) increases λ_2 and reduces over-squashing. DiffPool and GraphTransformer address this by learning global attention patterns.

Effective resistance. The effective resistance R_{ij} between nodes i and j (treating edges as unit resistors) equals $(\mathbf{e}_i - \mathbf{e}_j)^\top \mathbf{L}^\dagger (\mathbf{e}_i - \mathbf{e}_j)$, where $\mathbf{L}^\dagger$ is the pseudoinverse of $\mathbf{L}$. High effective

resistance between two nodes signals a bottleneck in information flow and is a principled criterion for graph rewiring.

Example — Spectral gap of cycle vs. complete graph on $N = 6$ nodes

For the 6-cycle C_6: eigenvalues of $\hat{\mathbf{L}}$ are $\lambda_k = 1 - \cos(2\pi k/6)$ for $k = 0, \dots, 5$. Thus $\lambda_2 = 1 - \cos(2\pi/6) = 1 - 1/2 = 0.5$.
For the complete graph K_6: $\hat{\mathbf{L}} = \frac{6}{5}(\mathbf{I} - \frac{1}{6}\mathbf{J})$, so $\lambda_2 = \frac{6}{5}$ (all non-zero eigenvalues equal). Spectral gap $\lambda_2(K_6)/\lambda_2(C_6) = 2.4$: the complete graph mixes $2.4\times$ faster than the cycle, consistent with diameter 1 vs. diameter 3.

25.5 Random Walk Graph Kernels

I *graph kernels* misurano la similarità tra grafi interi, utili per classificazione di molecole, reti proteiche, ecc.

Random Walk Kernel. Definisce un prodotto diretto $\mathcal{G}_\times = \mathcal{G}_1 \times \mathcal{G}_2$ (nodi = coppie di nodi, archi = coppie di archi). La matrice di transizione del prodotto è $\mathbf{W}_\times = \mathbf{W}_1 \otimes \mathbf{W}_2$ (prodotto di Kronecker). Il kernel conta i cammini che i due grafi condividono:

$$k(\mathcal{G}_1, \mathcal{G}_2) = \sum_{t=0}^{\infty} \lambda^t \sum_{i,j} [\mathbf{W}_\times^t]_{ij} = \mathbf{1}^\top (I - \lambda \mathbf{W}_\times)^{-1} \mathbf{1}$$

con $\lambda < 1/\rho(\mathbf{W}_\times)$ per garantire convergenza.

Weisfeiler-Lehman (WL) Kernel. Algoritmo iterativo di raffinamento dei colori:

1. $c_i^{(0)}$ = colore iniziale (es. label del nodo).
2. $c_i^{(t+1)} = \text{hash}\big(c_i^{(t)}, \{\{c_j^{(t)} : j \in \mathcal{N}(i)\}\}\big)$
3. Feature vector = istogramma dei colori; kernel = prodotto interno.

Complessità: $O(Nh)$ con h iterazioni. Connessione con GNN: il WL test è esattamente il meccanismo di aggregazione di un GNN senza parametri learnable; il potere espressivo di GNN standard è limitato superiormente dal WL test (Xu et al. 2019).

Example — Limiti del WL test

I grafi regolari d-regolari di dimensioni diverse possono avere la stessa firma WL ma diverse proprietà strutturali (es. cicli, sottografi). I GNN standard non possono distinguerli. Soluzioni: GIN con somma invece di media; PPGN con cammini di ordine superiore; Random Features (RWSE).

Graph kernels define a similarity measure $k(G, G')$ between graphs, enabling kernel methods (SVMs, GPs) on graph-structured data.

Random walk kernel. Count pairs of matching walks of length l in G and G':

$$k_{\text{rw}}(G, G') = \sum_{l=0}^{\infty} \lambda^l \langle \mathbf{p}^{G,l}, \mathbf{p}^{G',l} \rangle$$

where $\mathbf{p}^{G,l}$ is the distribution over walk endpoints of length l starting from a fixed node in G.

Weisfeiler-Lehman (WL) subtree kernel. Iteratively refines node labels by aggregating neighbour labels (identical to the WL graph isomorphism test). Computes kernel by counting matching subtree patterns. This is precisely the computation performed by GNNs — the WL kernel serves as an expressiveness upper bound for 1-dimensional GNNs.

Shortest-path kernel. Another important graph kernel counts pairs of shortest paths of the same length and endpoint labels:

$$k_{\text{sp}}(G, G') = \sum_{v,v'} \sum_{u,u'} \mathbf{1}[d_G(v, v') = d_{G'}(u, u')] \cdot k_{\text{node}}(l_v, l_u) \cdot k_{\text{node}}(l_{v'}, l_{u'})$$

Computable in $O(N^3)$ via Floyd-Warshall. The rich zoo of graph kernels — random walk, WL, shortest-path, Graphlet — each captures different aspects of graph structure, making the choice of kernel a form of inductive bias.

PART 26

Spectral Graph Convolution and ChebNet

We derive spectral convolution, ChebNet's polynomial approximation, and show how spatial graph convolution emerges as a special case.

26.1 Spectral Convolution

Partendo dalla convoluzione spettrale $\mathbf{y} = \mathbf{U}\, g_\theta(\square)\, \mathbf{U}^\top \mathbf{x}$ con filtro $g_\theta(\square) = \mathrm{diag}(\theta_1, \dots, \theta_N)$, il problema principale è la complessità $O(N^2)$ e la non-localizzazione.

Filtri come funzioni del Laplaciano. Se $g_\theta(\square) = p(\square)$ è un polinomio di grado K:

$$\mathbf{U}\, p(\square)\, \mathbf{U}^\top = p(\mathbf{U}\square\mathbf{U}^\top) = p(\tilde{\mathbf{L}})$$

Poiché $\tilde{\mathbf{L}}^k_{ij} = 0$ se $d(i,j) > k$ (distanza grafo-teorica), il filtro polinomiale di grado K è *K-localizzato*: aggrega solo informazione dal K-hop neighborhood. Complessità di applicazione: $O(KM)$ (sparse matrix-vector, $M = |\mathcal{E}|$).

Numero di parametri. Un filtro polinomiale $p(\lambda) = \sum_{k=0}^{K} \theta_k \lambda^k$ ha $K+1$ parametri *scalari*, indipendentemente da N. Questo consente la trasferibilità tra grafi di dimensioni diverse.

Proprietà di localizzazione-frequenza. C'è un trade-off: polinomi di grado basso hanno buona localizzazione spaziale (piccolo K-hop) ma scarsa selettività in frequenza; polinomi di grado alto viceversa (analogo al principio di indeterminazione di Heisenberg per la STFT).

A spectral graph convolution applies a filter g_θ in the frequency domain:

$$\mathbf{y} = g_\theta(\mathbf{L})\mathbf{x} = \mathbf{U}\, g_\theta(\square)\, \mathbf{U}^\top \mathbf{x}$$

where $g_\theta(\square) = \mathrm{diag}(g_\theta(\lambda_1), \dots, g_\theta(\lambda_N))$.

Problems: (1) $O(N^2)$ to apply $\mathbf{U}$; (2) filter not localised in space; (3) $\mathbf{U}$ changes with graph.

Band-pass and low-pass filters on graphs. Just as in classical signal processing, one can design:

- *Low-pass filter*: $g_\theta(\lambda) = e^{-\tau\lambda}$ (heat kernel). Suppresses high frequencies (oscillatory, non-smooth signals), performing smoothing. Equivalent to running the graph heat equation for time τ.
- *High-pass filter*: $g_\theta(\lambda) = \lambda$ (the Laplacian itself). Amplifies high-frequency components, detecting edges/discontinuities in the signal.
- *Band-pass*: combination of polynomial terms selecting intermediate frequencies.

EXAMPLE — Heat kernel filter on a graph

For the star graph S_5 (one centre node, 4 leaves) with $\hat{\mathbf{L}}$ normalised Laplacian: eigenvalues are $\{0, 1, 1, 1, 2\}$. Applying heat kernel $g(\lambda) = e^{-0.5\lambda}$:

$$g(0) = 1.0, \quad g(1) = e^{-0.5} \approx 0.607, \quad g(2) = e^{-1} \approx 0.368$$

The zero-frequency component (uniform signal) passes unchanged; the highest frequency is attenuated by 63%. After applying this filter, signals are smoother across edges.

Example — ChebNet vs GCN: a concrete comparison

For a graph with $n = 1000$ nodes and $k = 3$ Chebyshev polynomials: ChebNet computes $\mathbf{K} \in \mathbb{R}^{1000\times 1000}$ as $\{T_0, T_1, T_2\}$ of $\tilde{\mathbf{L}}$, requiring 3 sparse matrix-vector products per layer. GCN approximates with $k = 1$ (first-order), using only $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ — one sparse multiplication. On Cora: ChebNet with $k = 3$, $d = 64$ achieves 81.2% accuracy; GCN with $k = 1$, $d = 64$ achieves 81.5% with $3\times$ faster training. The gain from higher Chebyshev order saturates quickly on small-world graphs where information propagates efficiently even with $k = 1$.

Spectral vs. spatial convolution equivalence For the polynomial filter $h(\mathbf{L}) = \sum_{k=0}^{K} \theta_k \mathbf{L}^k$, the output $h(\mathbf{L})\mathbf{f}$ at node v depends only on nodes within K-hop neighbourhood of v: $[h(\mathbf{L})\mathbf{f}]_v = \sum_{u:\, d(u,v)\le K} \alpha_{vu} f_u$ for some coefficients α_{vu}. This equivalence between polynomial spectral filters and localised spatial convolutions justifies using the Chebyshev polynomial basis (ChebNet) or the first-order approximation (GCN) as computationally efficient substitutes for the full spectral decomposition.

26.2 ChebNet: Polynomial Approximation

Deffrrard et al. (2016) propongono di usare i *polinomi di Chebyshev* T_k come base per i filtri, per sfruttarne l'ottimalità di approssimazione in norma L^∞.

Polinomi di Chebyshev. $T_0(x) = 1$, $T_1(x) = x$, $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$. Ortonormali su $[-1, 1]$ rispetto a $w(x) = (1 - x^2)^{-1/2}$.

ChebNet filter. Si riscalano gli autovalori in $[-1, 1]$: $\tilde{\lambda}_k = 2\lambda_k/\lambda_{\text{max}} - 1$. Il filtro è:

$$g_\theta(\mathbf{L}) = \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{L}}), \qquad \tilde{\mathbf{L}} = \frac{2}{\lambda_{\text{max}}}\mathbf{L} - \mathbf{I}$$

I termini $T_k(\tilde{\mathbf{L}})\mathbf{x}$ si calcolano per ricorrenza in $O(K|\mathcal{E}|)$:

$$\bar{\mathbf{x}}_0 = \mathbf{x}, \quad \bar{\mathbf{x}}_1 = \tilde{\mathbf{L}}\mathbf{x}, \quad \bar{\mathbf{x}}_k = 2\tilde{\mathbf{L}}\bar{\mathbf{x}}_{k-1} - \bar{\mathbf{x}}_{k-2}$$

Propagation layer. Per F_{in} feature in input e F_{out} in output:

$$\mathbf{Z} = \sum_{k=0}^{K} \bar{\mathbf{X}}_k \square_k, \qquad \square_k \in \mathbb{R}^{F_{\text{in}}\times F_{\text{out}}}$$

Totale parametri per layer: $(K + 1)F_{\text{in}}F_{\text{out}}$.

Example — Scelta di K

$K = 1$: filtro lineare in $\mathbf{L}$, corrisponde a GCN di Kipf&Welling ($K = 1$, $\lambda_{\text{max}} = 2$ approssimato). $K = 2$: aggrega 2-hop, spesso sufficiente per task di classificazione su citation networks. $K > 6$: raramente usato per over-smoothing.

Approximate $g_\theta(\square)$ with a degree-K Chebyshev polynomial:

$$g_\theta(\square) \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\square})$$

where $\tilde{\Box} = \frac{2\Box}{\lambda_{\max}} - \mathbf{I}$ (rescaled to $[-1, 1]$) and T_k are Chebyshev polynomials of the first kind:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$

Since $T_k(\tilde{\mathbf{L}}) = \mathbf{U}T_k(\tilde{\Box})\mathbf{U}^\top$ (spectral mapping theorem), the convolution becomes:

$$\mathbf{y} = \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathbf{L}})\mathbf{x}$$

Key: this avoids eigendecomposition entirely. Computing $T_k(\tilde{\mathbf{L}})\mathbf{x}$ uses the recurrence, each step a sparse matrix-vector multiply: $O(K|\mathcal{E}|)$ total. The K-hop receptive field: node i aggregates information from all nodes within K hops.

Why Chebyshev and not monomials? Polynomial interpolation using monomials $\{1, x, x^2, ...\}$ in the basis suffers from numerical ill-conditioning: the Vandermonde system has exponentially large condition number. Chebyshev polynomials, being orthogonal on $[-1, 1]$, avoid this. Moreover, Chebyshev expansion minimises the maximum approximation error (minimax property): for any continuous g on $[-1, 1]$, the degree-K Chebyshev approximation is near-optimal in L^∞ norm.

Example — Chebyshev approximation of low-pass filter

Approximate the ideal low-pass $g(\lambda) = \mathbf{1}[\lambda \leq 0.5]$ with $K = 4$ Chebyshev terms on $[-1, 1]$ (rescaled from $[0, 2]$). The Chebyshev coefficients are $\theta_k = \frac{2}{\pi}\int_{-1}^{1} g(\lambda)T_k(\lambda)(1-\lambda^2)^{-1/2}d\lambda$:

$$\theta_0 = 0.5, \ \theta_1 = \frac{1}{\pi}, \ \theta_2 = 0, \ \theta_3 = -\frac{1}{3\pi}, \ \theta_4 = 0$$

This 4-term filter approximates the step function with maximum error ≈ 0.18. With $K = 10$, the error drops to 0.06 (Gibbs-like oscillations near the discontinuity).

26.3 Graph Convolutional Network (GCN)

Kipf & Welling (2017) semplificano ChebNet con $K = 1$ e l'approssimazione $\lambda_{\max} \approx 2$:

$$g_\theta * \mathbf{x} \approx \theta_0\mathbf{x} + \theta_1(\mathbf{L} - \mathbf{I})\mathbf{x} = \theta_0\mathbf{x} - \theta_1\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{x}$$

Con il vincolo $\theta = \theta_0 = -\theta_1$ (riduzione a 1 parametro per evitare overfitting):

$$g_\theta * \mathbf{x} \approx \theta(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}$$

La matrice $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ ha autovalori in $[0, 2]$. Per evitare instabilità numerica con layer profondi, *renormalization trick*: $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{A}_{ij}$:

GCN layer (forma finale)

$$\mathbf{H}^{(\ell+1)} = \sigma\left(\hat{\mathbf{A}}\,\mathbf{H}^{(\ell)}\,\mathbf{W}^{(\ell)}\right), \qquad \hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$$

dove $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ è la matrice dei pesi.

Interpretazione messaggi. Per il nodo i:

$$h_i^{(\ell+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)\cup\{i\}} \frac{1}{\sqrt{d_i d_j}} \mathbf{W}^{(\ell)} h_j^{(\ell)}\right)$$

Media pesata dai gradi: nodi con molti vicini contribuiscono meno (normalizzazione simmetrica).

Over-smoothing. Con L layer profondi, $\mathbf{H}^{(L)} \approx \mathbf{U}\square^{L}\mathbf{U}^{\top}\mathbf{H}^{(0)}\mathbf{W}$. I coefficienti delle frequenze alte vengono smorzati esponenzialmente; tutti i nodi tendono alla stessa rappresentazione (la componente DC). Soluzioni: residual connections, DropEdge, PairNorm, decoupling di profondità e ampiezza.

Complessità. Pre-calcolo $\hat{\mathbf{A}}$: $O(M)$. Forward pass per layer: $O(Md)$ (sparse-dense product). Totale per L layer: $O(LMd)$.

Kipf & Welling (2017) take $K = 1$ and approximate $\lambda_{\max} \approx 2$, giving $\tilde{\mathbf{L}} \approx \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$. With the "renormalisation trick" $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{A}_{ij}$:

$$\mathbf{H}^{(l+1)} = \sigma\big(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\big)$$

Spatial interpretation. The propagation rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \sum_{j \in \mathcal{N}(i)\cup\{i\}} \frac{1}{\sqrt{d_i d_j}} \mathbf{h}_j^{(l)}\right)$$

Node i aggregates normalised neighbours plus itself, then applies a shared linear transform.

Example — GCN propagation on a triangle graph

Nodes $\{1, 2, 3\}$, fully connected. $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ gives all-ones matrix. $\tilde{\mathbf{D}} = \mathrm{diag}(3, 3, 3)$.
$\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2} = \frac{1}{3}\mathbf{J}$ (where $\mathbf{J}$ is the all-ones matrix).
For input $\mathbf{H}^{(0)} = (1, 0, 0; 0, 1, 0; 0, 0, 1) = \mathbf{I}$, weight $\mathbf{W}^{(0)} = \mathbf{I}$:

$$\mathbf{H}^{(1)} = \sigma\left(\frac{1}{3}\mathbf{J}\right) = \sigma\left(\begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}\right)$$

All nodes converge to the same representation — over-smoothing after too many GCN layers.

Connection to label propagation. The GCN propagation matrix $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$ is closely related to the label propagation algorithm used in semi-supervised learning. Label propagation iterates $\mathbf{F}^{(t+1)} = \alpha\hat{\mathbf{A}}\mathbf{F}^{(t)} + (1-\alpha)\mathbf{Y}$ to convergence, where $\mathbf{Y}$ are the observed labels and α controls the influence of neighbours. GCN can be seen as a learnable, non-linear extension of this classical algorithm.

26.4 Graph Isomorphism Network (GIN)

Xu et al. (2019) caratterizzano il potere espressivo dei GNN confrontandolo con il WL graph isomorphism test (1-WL).

Teorema (Xu et al. 2019). Qualsiasi GNN con aggregazione iniettiva è al più potente quanto 1-WL. Un GNN raggiunge questo massimo sse la funzione di aggregazione è iniettiva nel multiset dei vicini.

Funzioni iniettive su multiset. Per multiset X su un universo numerabile, la funzione $f(X) = (1+\varepsilon)\,h(v) + \sum_{u\in X} h(u)$ è iniettiva per ε non intero, con h iniettiva. Approssimata con MLP:

GIN layer

$$h_v^{(k)} = \text{MLP}^{(k)}\left((1 + \varepsilon^{(k)})\, h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)} \right)$$

dove $\varepsilon^{(k)}$ può essere learnable o fissato a 0.

Perché somma e non media/max? Media: non distingue multiset con stesso valor medio ma diversi elementi. Max: non distingue multiset con stesso massimo. Somma: iniettiva per multiset di cardinalità variabile su distribuzioni con supporto numerabile.

Example — GIN vs GCN su strutture distinguibili

I grafi $\mathcal{G}_1$ (6-ciclo) e $\mathcal{G}_2$ (due triangoli disgiunti) hanno la stessa firma WL al primo livello con GCN (media), ma diversa con GIN (somma): $\mathcal{G}_1$ ha un nodo con 2 vicini di grado 2 ciascuno nel vicinato; $\mathcal{G}_2$ ha lo stesso ma il conteggio totale differisce. GIN li distingue al 2° layer.

Graph-level readout iniettivo. $h_{\mathcal{G}=\text{MLP}\left(\sum_{k=0}^{K}\sum_{v\in\mathcal{V}} h_v^{(k)}\right)}$, concatenando rappresentazioni di tutti i layer (JK-connection).

Xu et al. (2019) prove that the most expressive GNN aggregation (among 1-WL equivalent methods) must use an injective function:

$$\mathbf{h}_v^{(k)} = \text{MLP}^{(k)}\left((1 + \epsilon)\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k-1)} \right)$$

where ϵ is a learnable parameter. The $(1+\epsilon)\mathbf{h}_v$ term distinguishes the central node from its neighbours.

Why sum aggregation? Sum is injective on multisets (unlike mean or max, which can conflate different neighbourhood distributions). For example:

- Mean aggregation: cannot distinguish a node with 2 neighbours both of feature 1 from one with 4 neighbours both of feature 1.
- Sum: $1 + 1 = 2$ vs $1 + 1 + 1 + 1 = 4$ — distinguishable.

Graph-level readout. Pool node representations across layers (not just the last layer) using a differentiable aggregation:

$$\mathbf{h}_G = \text{CONCAT}\Big(\text{READOUT}\Big(\{\mathbf{h}_v^{(k)}\}_{v\in G}\Big),\ k = 0, 1, \ldots, K\Big)$$

This captures both local and global features.

Beyond 1-WL: higher-order GNNs. The 1-WL test assigns colours to individual nodes; the k-WL test ($k \geq 2$) assigns colours to k-tuples of nodes and is strictly more powerful. Networks operating on k-tuples (PPGN, k-GNN) can distinguish graphs that 1-WL cannot, at the cost of $O(N^k)$ complexity. For molecular property prediction, 3-WL equivalent networks (e.g. HIMP) achieve state-of-the-art results on benchmarks where 1-WL fails.

PART 27

Graph Attention Networks and GraphSAGE

Attention mechanisms allow nodes to weight their neighbours differently. GraphSAGE enables inductive learning on unseen graphs.

27.1 Graph Attention Network (GAT)

Veličković et al. (2018) introducono meccanismi di attention per pesare diversamente i vicini, senza dipendere dalla struttura del grafo (non come GCN).

Calcolo dell'attention. Per la coppia di nodi (i, j) con $j \in \mathcal{N}(i)$:

$$e_{ij} = \text{LeakyReLU}\big(\mathbf{a}^\top \left[\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_j\right]\big)$$

dove $\mathbf{W} \in \mathbb{R}^{F' \times F}$, $\mathbf{a} \in \mathbb{R}^{2F'}$, $\|$ = concatenazione. Normalizzazione softmax sul vicinato:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(e_{ik})}$$

GAT layer

$$\mathbf{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right)$$

Con multi-head attention (K teste):

$$\mathbf{h}'_i = \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_j\right)$$

Complessità. $O(|\mathcal{V}|FF' + |\mathcal{E}|F')$ per layer con una testa. L'attention è calcolata edge-wise, quindi scala con $|\mathcal{E}|$ (efficiente per grafi sparsi).

GATv2 (Brody et al. 2022). Il GAT originale soffre di *static attention*: e_{ij} dipende da $\mathbf{h}_i$ e $\mathbf{h}_j$ separatamente prima della combinazione non-lineare, rendendolo di rango 1 nel calcolo dell'attention. GATv2 corregge:

$$e_{ij} = \mathbf{a}^\top \text{LeakyReLU}\big(\mathbf{W}[\mathbf{h}_i \,\|\, \mathbf{h}_j]\big)$$

ottenendo attention *dinamica* di rango pieno.

Interpretabilità. I pesi α_{ij} possono essere visualizzati per interpretare quale vicino contribuisce maggiormente all'embedding del nodo i, utile in applicazioni scientifiche (quali atomi influenzano un certo atomo in una molecola).

GCN uses fixed normalisation $1/\sqrt{d_i d_j}$ — every neighbour contributes equally (up to degree). GAT learns attention coefficients:

GAT Propagation

$$\alpha_{ij} = \frac{\exp\big(\text{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j])\big)}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LeakyReLU}(\mathbf{a}^\top[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_k]))}$$

$$\mathbf{h}_i' = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right)$$

where $\mathbf{a} \in \mathbb{R}^{2d'}$ and $\mathbf{W} \in \mathbb{R}^{d' \times d}$ are learned, $\|$ denotes concatenation.

Multi-head GAT. Run H independent attention heads and concatenate (or average):

$$\mathbf{h}_i' = \Big\|_{h=1}^{H} \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(h)} \mathbf{W}^{(h)} \mathbf{h}_j\right)$$

Complexity. $O(|\mathcal{V}|dd' + |\mathcal{E}|d')$ per layer — linear in graph size.

Relationship to transformer self-attention. If the graph is fully connected (all pairs of nodes), GAT reduces exactly to transformer self-attention. This observation motivates Graph Transformers: add global attention (all-pairs) to a GNN backbone to capture long-range dependencies without increasing depth. The Graph Transformer (Dwivedi & Bresson, 2021) and GPS (Rampášek et al., 2022) formalise this hybrid approach.

Example — GAT attention in citation networks

In the Cora citation network (2708 papers, 5429 edges, 7 classes), a 2-layer GAT achieves 83.0% test accuracy vs. 81.5% for GCN. Visualising α_{ij} on correctly classified nodes: papers tend to attend most strongly ($\alpha > 0.3$) to neighbours in the same category, with cross-category citations receiving lower weights. This interpretability advantage is lost in GCN where normalisation is fixed.

27.2 GraphSAGE: Inductive Learning on Large Graphs

Hamilton et al. (2017) introduced GraphSAGE (**S**ample and **AGG**regate) specifically to address two limitations of spectral GNNs: (1) transductive learning requiring the full graph Laplacian, and (2) $O(|\mathcal{V}|)$ memory and computation for each forward pass regardless of the mini-batch size.

Core idea: localise and generalise. A spectral GCN layer computes $\mathbf{H}' = \sigma(\hat{\mathbf{A}}\mathbf{H}\mathbf{W})$, which requires the full adjacency $\hat{\mathbf{A}} \in \mathbb{R}^{n \times n}$. GraphSAGE replaces this with a local aggregation function applied to a sampled neighbourhood $\tilde{\mathcal{N}}(v) \subset \mathcal{N}(v)$:

$$\mathbf{h}_v^{(k)} = \sigma\Big(\mathbf{W}^{(k)}\Big[\mathbf{h}_v^{(k-1)} \Big\| \operatorname{AGG}\big(\{\mathbf{h}_u^{(k-1)} : u \in \tilde{\mathcal{N}}(v)\}\big)\Big]\Big)$$

Because the aggregation is over a *multiset* of neighbours rather than a fixed matrix column, the weights $\mathbf{W}^{(k)}$ can be applied to any node at test time — including nodes not seen during training.

Inductive capability: formal argument. For a new node v^* with features $\mathbf{x}_{v^*}$ and neighbours $\mathcal{N}(v^*)$, compute $\mathbf{h}_{v^*}^{(k)}$ by applying the same aggregation and update functions learned during training. No retraining, no spectral re-decomposition. This works as long as the neighbourhood features $\{\mathbf{x}_u\}$ are available — satisfied for any node added to the graph with its attributes.

Unsupervised training. When node labels are unavailable, GraphSAGE can be trained with a graph-based loss:

$$\mathcal{L} = -\log \sigma(\mathbf{h}_v^\top \mathbf{h}_u) - Q\, \mathbb{E}_{u_n \sim P_n}\left[\log \sigma(-\mathbf{h}_v^\top \mathbf{h}_{u_n})\right]$$

Minimise co-embedded distance for nodes u co-occurring in random walks of length l from v (positive pairs); maximise distance from Q negatively-sampled nodes $u_n \sim P_n(u) \propto d_u^{3/4}$. This is analogous to the skip-gram objective in word2vec, adapted to graph random walks.

PinSage: GraphSAGE at Pinterest Scale

Ying et al. (2018) deployed GraphSAGE on Pinterest's graph (3B nodes, 18B edges) for item recommendation. Innovations: (1) *importance-based sampling* — sample neighbours proportional to random-walk visit frequency, prioritising structurally important nodes; (2) *curriculum training* — progressively harder negative examples during training (easy random negatives early, then hard negatives from the top-500 ranked items); (3) *MapReduce inference* — produce embeddings for all 3B items in batches across a cluster, enabling offline indexing for retrieval. PinSage reduced recommendation latency by $\sim 30\%$ and improved engagement metrics by $> 15\%$ vs. prior matrix-factorisation approach.

27.3 GraphSAGE: Sampling and Aggregation

GraphSAGE (Hamilton et al., 2017) addresses a key limitation of spectral GNNs: they require the full graph Laplacian at training time, making them transductive (unable to generalise to unseen nodes). GraphSAGE is designed for *inductive* learning on large graphs via neighbourhood sampling.

Neighbourhood sampling. Rather than aggregating all neighbours (which may number in the thousands for hub nodes), GraphSAGE samples a fixed-size neighbourhood at each layer:

$$\tilde{\mathcal{N}}^k(v) \sim \text{Uniform}(\mathcal{N}(v),\, S_k)$$

where S_k is the sample size at layer k (typically $S_1 = 25$, $S_2 = 10$). This bounds the computation per node to $\prod_k S_k$ neighbours regardless of actual degree.

Aggregation options. The aggregation function AGG can be:

- *Mean*: $\text{AGG}(\{\mathbf{h}_u\}) = \frac{1}{|\tilde{\mathcal{N}}(v)|}\sum_{u\in\tilde{\mathcal{N}}(v)} \mathbf{h}_u$ — equivalent to spectral GCN for full-batch.
- *Max-pool*: $\text{AGG}(\{\mathbf{h}_u\}) = \max_{u\in\tilde{\mathcal{N}}(v)} \sigma(\mathbf{W}_{\text{pool}}\mathbf{h}_u + \mathbf{b})$ — captures representative features.
- *LSTM*: apply an LSTM to a random permutation of $\tilde{\mathcal{N}}(v)$; permutation-non-invariant but empirically strong.

Update step. After aggregation, concatenate and project:

$$\mathbf{h}_v^{(k)} = \sigma\Big(\mathbf{W}^{(k)}\Big[\mathbf{h}_v^{(k-1)} \,\|\, \text{AGG}\Big(\{\mathbf{h}_u^{(k-1)} : u \in \tilde{\mathcal{N}}(v)\}\Big)\Big]\Big)$$

Normalise: $\mathbf{h}_v^{(k)} \leftarrow \mathbf{h}_v^{(k)} / \|\mathbf{h}_v^{(k)}\|_2$. Concatenation (rather than sum) preserves the node's own features independently of the aggregated message.

Mini-Batch Training

For a mini-batch of target nodes $\mathcal{B}$, define the computation graph recursively: at layer k, the required nodes are $\mathcal{B}_k = \mathcal{B}_{k+1} \cup \bigcup_{v\in\mathcal{B}_{k+1}} \tilde{\mathcal{N}}^k(v)$. The total nodes sampled is $|\mathcal{B}| \cdot \prod_k S_k$ — independent of the full graph size $|\mathcal{V}|$. This enables training on graphs with billions of nodes (e.g. Pinterest: 3B nodes, 18B edges) on commodity hardware.

Inductive setting. At test time, apply the learned aggregation functions to new nodes not seen during training — e.g. newly joined users in a recommendation graph. No retraining or matrix factorisation is needed; the model generalises by construction. This distinguishes GraphSAGE from DeepWalk, node2vec, and spectral GNNs.

> Neighbour Explosion in Deep GNNs With $K = 3$ layers and $S_1 = 25$, $S_2 = 10$, $S_3 = 5$, GraphSAGE samples up to $25 \cdot 10 \cdot 5 = 1250$ nodes per target. For mini-batches of $B = 512$ targets, this is 640K node feature lookups per step — feasible for $d = 128$ features but expensive for raw text or image features. *VR-GCN* (Chen et al., 2018) reduces this with a variance-reduction control variate: keep a historical activation cache $\bar{\mathbf{h}}_v$ for all nodes, and correct the stochastic estimate as $\widehat{\mathbf{h}}_v = \mathbf{h}_v^{\text{sample}} + \bar{\mathbf{h}}_v - \bar{\mathbf{h}}_v^{\text{sample}}$, reducing variance without increasing the neighbourhood sample size.

27.4 Heterogeneous Graphs and HAN

Real-world graphs often have multiple node types (users, movies, genres) and edge types (watches, rates, similar-to). A **heterogeneous information network (HIN)** is a graph $G = (\mathcal{V}, \mathcal{E})$ with type functions $\phi : \mathcal{V} \to \mathcal{A}$ and $\psi : \mathcal{E} \to \mathcal{R}$, where $|\mathcal{A}| + |\mathcal{R}| > 2$.

Meta-paths. A meta-path $\Phi = A_1 \xrightarrow{r_1} A_2 \xrightarrow{r_2} \cdots \xrightarrow{r_l} A_{l+1}$ defines a composite relation between node types. Example in a movie graph: User $\xrightarrow{\text{rates}}$ Movie $\xrightarrow{\text{directed-by}}$ Director $\xrightarrow{\text{directs}}$ Movie captures users sharing a preferred director.

Heterogeneous Graph Attention Network (HAN) (Wang et al., 2019). For each meta-path Φ:

$$\mathbf{h}_v^{\Phi} = \sigma\left(\sum_{u \in \mathcal{N}^{\Phi}(v)} \alpha_{vu}^{\Phi} \, \mathbf{W}^{\Phi} \mathbf{h}_u \right), \qquad \alpha_{vu}^{\Phi} = \frac{\exp(\mathbf{a}^{\Phi\top} [\mathbf{W}^{\Phi}\mathbf{h}_v \,\|\, \mathbf{W}^{\Phi}\mathbf{h}_u])}{\sum_{w \in \mathcal{N}^{\Phi}(v)} \exp(\cdots)}$$

Then aggregate across meta-paths with learnable semantic attention weights $\{w_{\Phi}\}$:

$$\mathbf{h}_v = \sum_{\Phi} w_{\Phi} \, \mathbf{h}_v^{\Phi}, \qquad w_{\Phi} = \frac{\exp\big(\mathbf{q}^{\top} \tanh(\mathbf{W}\bar{\mathbf{h}}^{\Phi} + \mathbf{b})\big)}{\sum_{\Phi'} \exp(\cdots)}$$

where $\bar{\mathbf{h}}^{\Phi} = \frac{1}{|\mathcal{V}|} \sum_v \mathbf{h}_v^{\Phi}$ is the mean meta-path embedding.

Relational GCN (R-GCN) for Knowledge Graphs

Knowledge graphs have typed edges $r \in \mathcal{R}$ (e.g. `bornIn`, `worksAt`). R-GCN (Schlichtkrull et al., 2018) propagates per-relation:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)} \mathbf{h}_i^{(l)} + \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_r(i)} \frac{1}{c_{i,r}} \mathbf{W}_r^{(l)} \mathbf{h}_j^{(l)} \right)$$

with $c_{i,r} = |\mathcal{N}_r(i)|$. Separate weight matrices $\mathbf{W}_r$ per relation capture different semantics; *basis decomposition* $\mathbf{W}_r = \sum_b a_{rb} \mathbf{V}_b$ reduces parameters from $|\mathcal{R}| \cdot d^2$ to $(B + |\mathcal{R}|) \cdot d^2$.

HGT: Heterogeneous Graph Transformer (Hu et al., 2020). Rather than meta-paths, HGT

uses type-specific projections within a Transformer-style attention:

$$\text{Attn}(s,e,t) = \underset{\blacksquare\{\text{z}\blacksquare\}}{\mathbf{h}_s \mathbf{K}_s^{\phi\top}} \cdot \underset{\blacksquare\{\text{z}\blacksquare\}}{\mathbf{W}_{\text{ATT}}^{\psi(e)}} \cdot \underset{\blacksquare\{\text{z}\blacksquare\}}{\mathbf{Q}_t^{\phi_t} \mathbf{h}_t}$$
$$\text{source} \qquad \text{edge} \qquad \text{target}$$

where ϕ_s, ϕ_t are source/target node types and $\psi(e)$ is the edge type. HGT scales to web-scale graphs (OAG: 179M nodes, 2.8B edges).

Applications. Heterogeneous GNNs excel at: recommendation systems (user-item-attribute graphs), biomedical knowledge graphs (gene-protein-disease), academic citation networks (paper-author-venue), and fraud detection (account-transaction-merchant networks with typed relations).

Example — R-GCN for Knowledge Graph Completion

On FB15k-237 (14.5K entities, 237 relations), R-GCN with basis decomposition ($B = 2$ bases) achieves MRR $= 0.153$ on the entity-prediction task, comparable to TransE and DistMult. The key advantage: R-GCN jointly trains entity embeddings and relation-specific transformations, enabling generalisation to unseen relation compositions not present in training triples.

28.1 Groups and Symmetries

Gruppo

Un gruppo $(G, \cdot)$ è un insieme G con operazione binaria $\cdot : G \times G \to G$ soddisfacente:

- *Chiusura*: $g \cdot h \in G$ per tutti $g, h \in G$.
- *Associatività*: $(g \cdot h) \cdot k = g \cdot (h \cdot k)$.
- *Elemento neutro*: $\exists e \in G : e \cdot g = g \cdot e = g$.
- *Inverso*: $\forall g \exists g^{-1} : g \cdot g^{-1} = e$.

Esempi fondamentali.

- $(\mathbb{Z}, +)$: traslazioni intere.
- S_N: gruppo simmetrico di permutazioni di N elementi. $|S_N| = N!$.
- $SO(2)$: rotazioni del piano. $|SO(2)| = \infty$ (gruppo di Lie compatto).
- $SE(3) = SO(3) \ltimes \mathbb{R}^3$: rotazioni + traslazioni 3D. Fondamentale per molecole.
- $E(3) = O(3) \ltimes \mathbb{R}^3$: include riflessioni (gruppo Euclideo).

Azioni di gruppo. Un'azione sinistra di G su uno spazio $\mathcal{X}$: $\rho : G \times \mathcal{X} \to \mathcal{X}$, con $\rho(e, x) = x$ e $\rho(g, \rho(h, x)) = \rho(g \cdot h, x)$. Notazione: $g \cdot x$.

Orbite e stabilizzatori. Orbita di x: $\mathcal{O}_x = \{g \cdot x : g \in G\}$. Stabilizzatore di x: $G_x = \{g \in G : g \cdot x = x\}$ (sottogruppo). Teorema delle orbite: $|\mathcal{O}_x| = |G|/|G_x|$.

Rilevanza per Deep Learning. Le CNN sono equivarianti per traslazione (il gruppo $\mathbb{Z}^2$). I GNN sono equivarianti per permutazione (S_N). Costruire modelli equivarianti per $SE(3)$ permette di apprendere rappresentazioni molecolari che si comportano correttamente sotto rotazione e traslazione, senza data augmentation.

Definition: Group

A group $(G, \cdot)$ is a set G with a binary operation $\cdot$ satisfying:

1. **Closure:** $g \cdot h \in G$ for all $g, h \in G$.
2. **Associativity:** $(g \cdot h) \cdot k = g \cdot (h \cdot k)$.
3. **Identity:** $\exists\, e \in G$ such that $e \cdot g = g \cdot e = g$.
4. **Inverse:** $\forall\, g \in G$, $\exists\, g^{-1}$ with $g \cdot g^{-1} = e$.

Key groups in deep learning:

Group	Elements	Action on $\mathbb{R}^d$
$\mathbb{Z}_n$	Cyclic rotations	Image rotation by $2\pi k/n$
$SO(2)$	2×2 rotation matrices	Rotation in the plane
$SO(3)$	3×3 rotation matrices	3D rotation
$SE(3)$	Rotations + translations	3D rigid motions
S_n	Permutations of n elements	Reordering nodes/tokens

Subgroups and quotient spaces. A subset $H \subseteq G$ is a subgroup if it is itself a group under the same operation. The left cosets $\{gH : g \in G\}$ partition G; when H is normal ($gH = Hg$ for all g), the cosets form the quotient group G/H. Example: $SO(3)/SO(2) \cong S^2$ (the 2-sphere), which appears in the theory of steerable CNNs operating on spherical data.

28.2 Invariance and Equivariance

Invarianza ed Equivarianza

Sia $\phi : \mathcal{X} \to \mathcal{Y}$ una funzione, G un gruppo con azioni $\rho_{\mathcal{X}}$ su $\mathcal{X}$ e $\rho_{\mathcal{Y}}$ su $\mathcal{Y}$:

- ϕ è *G-invariante* se $\phi(g \cdot x) = \phi(x)$ per tutti $g \in G, x \in \mathcal{X}$.
- ϕ è *G-equivariante* se $\phi(g \cdot x) = g \cdot \phi(x)$ per tutti $g \in G, x \in \mathcal{X}$.

Esempi canonici. Classificazione di immagini sotto traslazione: la previsione deve essere invariante ($f(T_a \mathbf{x}) = f(\mathbf{x})$). Segmentazione semantica: invariante nel label, equivariante nella mappa di features ($F(T_a \mathbf{x}) = T_a F(\mathbf{x})$). Dinamica molecolare: l'energia $E(\{r_i\})$ è $SE(3)$-invariante; le forze $\{-\nabla_{r_i} E\}$ sono $SE(3)$-equivarianti.

GNN e permutation equivariance. Sia $\pi \in S_N$ una permutazione dei nodi. Un GNN è equivariante per permutazione sse:

$$f(\mathbf{P}_\pi \mathbf{X}, \mathbf{P}_\pi \mathbf{A} \mathbf{P}_\pi^\top) = \mathbf{P}_\pi f(\mathbf{X}, \mathbf{A})$$

dove $\mathbf{P}_\pi$ è la matrice di permutazione. I layer GCN e GAT soddisfano questa proprietà perché aggregano su insiemi non ordinati.

Pooling invariante. Per ottenere una rappresentazione a livello di grafo (invariante per permutazione), si deve usare un'operazione invariante: somma, media, max sul multiset $\{h_v : v \in \mathcal{V}\}$. La somma è la scelta di default in GIN.

Universalità. Maron et al. (2019): le funzioni continue S_N-invarianti su tensori di nodi si possono approssimare arbitrariamente bene da reti invarianti per permutazione con operazioni tensore di ordine appropriato. Reti del 2° ordine (operando su coppie di nodi) sono strettamente più espressive di 1-WL.

Let G act on $\mathcal{X}$ via $\rho_{\mathcal{X}(g)}$ and on $\mathcal{Y}$ via $\rho_{\mathcal{Y}(g)}$.

Definitions: Invariance and Equivariance

A function $f : \mathcal{X} \to \mathcal{Y}$ is:

$$\textbf{Invariant:} \quad f(\rho_{\mathcal{X}}(g)\cdot \mathbf{x}) = f(\mathbf{x}) \quad \forall\, g \in G, \mathbf{x} \in \mathcal{X}$$

$$\textbf{Equivariant:} \quad f(\rho_{\mathcal{X}}(g)\cdot \mathbf{x}) = \rho_{\mathcal{Y}}(g)\cdot f(\mathbf{x}) \quad \forall\, g \in G, \mathbf{x} \in \mathcal{X}$$

Examples:

- A graph classifier should be *invariant* to node permutation: same label regardless of node ordering.
- A per-node embedding should be *equivariant*: permuting input nodes permutes the embeddings the same way.
- A 3D force predictor should be *equivariant* to $SO(3)$: rotating the molecule rotates the predicted forces.

Equivariance composition. If f is G-equivariant and g is G-equivariant, then $g \circ f$ is G-equivariant. Deep equivariant networks stack equivariant layers; a final invariant pooling gives an invariant global output.

Benefits of built-in symmetry. Encoding symmetry as an architectural constraint (rather than learning it from data) provides three concrete advantages:

1. *Sample efficiency*: the network never needs to learn that a rotated molecule is the same as the original — this is guaranteed by construction.
2. *Generalisation*: equivariant models generalise to unseen orientations without data augmentation.
3. *Parameter efficiency*: weight sharing across group orbits reduces the effective number of parameters by a factor of $|G|$.

28.3 Group Convolution (G-CNN)

Cohen & Welling (2016) extend standard convolutional networks to be equivariant under broader symmetry groups G (e.g. rotations, reflections) by replacing spatial convolution with *group convolution*.

Standard convolution as a special case. A 2D convolution can be written as:

$$(f \star \psi)(x) = \sum_{y \in \mathbb{Z}^2} f(y)\, \psi(y - x)$$

This is equivariant to translation $g \in \mathbb{Z}^2$: translating f by g translates the output by g. The key operation is indexing the filter by the group action $y - x = t_x^{-1} \cdot y$.

Group convolution. For a group G acting on a domain Ω, the group convolution of $f : \Omega \to \mathbb{R}^c$ by filter $\psi : G \to \mathbb{R}^c$ is:

$$(f \star_G \psi)(g) = \sum_{h \in G} f(h)\, \psi(g^{-1}h)$$

The output is now a function on G. In subsequent layers, both input and output are functions on G:

$$(f \star_G \psi)(g) = \sum_{h \in G} f(h)\, \psi(g^{-1} \cdot h)$$

Equivariance. $L_g f \star_G \psi = L_g(f \star_G \psi)$, where $(L_g f)(h) = f(g^{-1}h)$. The output feature map transforms the same way as the input under any group element.

Implementation for $p4$ (90° rotations + translations). For the group of $90°$ rotations (C_4) combined with translations ($\mathbb{Z}^2$), the group $G = p4$ has $4n^2$ elements for an $n \times n$ grid. The first layer takes $\mathbb{R}^{H\times W\times c_{\text{in}}}$ as input and produces $\mathbb{R}^{H\times W\times 4\times c_{\text{out}}}$ (one plane per rotation); subsequent layers operate on $\mathbb{R}^{H\times W\times 4\times c}$. The filter bank has $4\times$ fewer independent parameters than four separate non-equivariant filters.

Parameter sharing and efficiency

G-CNN achieves equivariance *without* augmentation, using the same number of parameters as a CNN without rotation-equivariance but with $4\times$ data augmentation. On CIFAR-10 and medical image segmentation tasks with rotational symmetry, G-CNNs match or exceed heavily-augmented CNNs at $4\times$ fewer training samples (Cohen & Welling, 2016).

Steerable CNNs. Weiler et al. (2019) generalise G-CNNs to continuous rotation groups $SO(2)$ and $SO(3)$ using irreducible representations (Wigner D-matrices). Feature maps carry *types* specifying how they transform under the group, and filters must intertwine the input and output types — a linear constraint that reduces the space of valid filters and enables analytical computation of equivariant bases.

Example — G-CNN on MNIST-rot

MNIST-rot contains 12K training images at random rotations. Standard CNN: 91.2% accuracy. Same CNN with $4\times$ augmentation: 96.1%. G-CNN with $p4$ group (4 rotations): 98.4%, using the same number of parameters as the non-augmented CNN. G-CNN with $p4m$ (4 rotations + reflections): 98.9%. The key result: G-CNN achieves better accuracy than the augmented CNN at no extra training cost — the symmetry is built in rather than learned from data.

28.4 Representations of Groups and Schur's Lemma

A **linear representation** of a group G is a group homomorphism $\rho : G \to GL(V)$ — a map assigning to each $g \in G$ an invertible linear map $\rho(g) : V \to V$ such that $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$.

Irreducible representations (irreps). A representation ρ on V is *irreducible* if there is no non-trivial G-invariant subspace $W \subsetneq V$ (i.e. $\rho(g)W \subseteq W \ \forall g \in G$ only for $W = \{0\}$ or $W = V$). The irreps are the "atoms" of representation theory: every finite-dimensional representation decomposes as a direct sum of irreps (Maschke's theorem for finite groups; Peter-Weyl for compact Lie groups):

$$V \cong \bigoplus_{\lambda \in \hat{G}} V_\lambda^{\oplus m_\lambda}$$

where $\hat{G}$ is the set of equivalence classes of irreps and m_λ are multiplicities.

Schur's Lemma

Let $\rho_1 : G \to GL(V_1)$ and $\rho_2 : G \to GL(V_2)$ be irreducible representations, and $\phi : V_1 \to V_2$ a linear map intertwining them: $\phi \circ \rho_1(g) = \rho_2(g) \circ \phi$ for all $g \in G$.

1. If $\rho_1 \not\cong \rho_2$, then $\phi = 0$.
2. If $\rho_1 = \rho_2 = \rho$ (and the field is $\mathbb{C}$), then $\phi = \lambda \mathrm{Id}$ for some $\lambda \in \mathbb{C}$.

Implications for equivariant networks. A linear map $\mathbf{W} : V_1 \to V_2$ between feature spaces is G-equivariant iff $\mathbf{W} \circ \rho_1(g) = \rho_2(g) \circ \mathbf{W}$ for all $g \in G$. By Schur's Lemma, in the irrep-decomposed basis such $\mathbf{W}$ is *block-diagonal*: it can only map irrep-λ features in V_1 to irrep-λ features in V_2, and within each block it is a scalar multiple of the identity (for the same irrep). This dramatically constrains the weight matrix: instead of $d_1 d_2$ free parameters, an equivariant linear layer has at most $\sum_\lambda m_{1,\lambda} m_{2,\lambda}$ free parameters (one scalar per pair of matching irrep multiplicities).

Characters. The *character* $\chi_\rho(g) = \mathrm{tr}(\rho(g))$ is a class function (constant on conjugacy classes). For finite groups, $\langle \chi_{\rho_1}, \chi_{\rho_2} \rangle = \delta_{\rho_1 \rho_2}$ (orthonormality), providing a practical tool for decomposing representations and verifying irreducibility without constructing explicit matrices.

Application to design. In e3nn (Geiger & Smidt, 2022), feature channels are typed by their $SO(3)$ irrep (degree $l \in \{0, 1, 2, ...\}$). Equivariant linear layers are scalings within each l-type; non-linearity is applied only to the $l = 0$ (scalar) channels; mixing across l-types uses the Clebsch-Gordan tensor product (§28.5). The number of free parameters grows as $O(c_{\text{max}}^2)$ rather than $O(c_{\text{max}}^2 \cdot (2l_{\text{max}} + 1)^2)$.

> Irreps of $SO(3)$ and their dimensions The irreducible representations of $SO(3)$ are indexed by non-negative integers $l = 0, 1, 2, ...$ with dimension $2l + 1$. For $l = 0$: trivial representation (scalars, invariant). For $l = 1$: standard 3D representation (vectors, equivariant under rotations). For $l = 2$: traceless symmetric tensors (dimension 5), relevant to quadrupole moments. The Peter-Weyl theorem guarantees $L^2(SO(3)) = \bigoplus_{l=0}^{\infty} V^{(l) \oplus (2l+1)}$: the Wigner D-matrices $(2l + 1)^2$ entries each form a complete orthogonal basis. For practical equivariant networks, truncation at $l_{\text{max}} = 2$ or 3 suffices for most molecular tasks.

28.5 The Wigner D-Matrix and Spherical Harmonics

Group representations for $SO(3)$ are labelled by a non-negative integer l (the *degree* or *angular momentum quantum number*). The irreducible representations (irreps) of $SO(3)$ are $(2l + 1)$-dimensional and are described by the **Wigner D-matrices**.

Wigner D-matrix. For a rotation $R \in SO(3)$ parametrised by Euler angles (α, β, γ), the degree-l Wigner matrix is:

$$D^{(l)}_{mm'}(R) = D^{(l)}_{mm'}(\alpha, \beta, \gamma) = e^{-im\alpha}\, d^{(l)}_{mm'}(\beta)\, e^{-im'\gamma}$$

where $d^{(l)}_{mm'}(\beta)$ are the *Wigner small d-matrices* (real-valued, computable from Jacobi polynomials), and $m, m' \in \{-l, \dots, l\}$. The matrices $\{D^{(l)}\}_{l=0,1,2,\dots}$ form a complete orthogonal basis for $L^2(SO(3))$ — the Peter-Weyl theorem.

Spherical harmonics. The restriction of $D^{(l)}$ to functions on $S^2 \cong SO(3)/SO(2)$ gives the real spherical harmonics $Y_l^m : S^2 \to \mathbb{R}$. For a unit vector $\hat{\mathbf{r}} = (\theta, \phi)$:

$$Y_l^m(\hat{\mathbf{r}}) = \sqrt{\frac{(2l+1)(l-|m|)!}{4\pi(l+|m|)!}}\, P_l^{|m|}(\cos\theta)\, e^{im\phi}$$

where P_l^m are associated Legendre polynomials. Key properties:

- *Orthonormality*: $\int_{S^2} Y_l^m(\hat{\mathbf{r}}) \overline{Y_{l'}^{m'}}(\hat{\mathbf{r}})\, d\hat{\mathbf{r}} = \delta_{ll'}\delta_{mm'}$.
- *Equivariance*: under rotation $R \in SO(3)$, $Y_l^m(R^{-1}\hat{\mathbf{r}}) = \sum_{m'} D^{(l)}_{mm'}(R)\, Y_l^{m'}(\hat{\mathbf{r}})$. A degree-$l$ spherical harmonic channel transforms according to the $D^{(l)}$ irrep.

- *Addition theorem*: $\sum_m Y_l^m(\hat{\mathbf{r}})Y_l^m(\hat{\mathbf{r}}') = \frac{2l+1}{4\pi}P_l(\hat{\mathbf{r}}\cdot\hat{\mathbf{r}}')$.

Clebsch-Gordan Decomposition

The tensor product of two irreps decomposes into a direct sum of irreps:

$$V^{(l_1)} \otimes V^{(l_2)} \cong \bigoplus_{l=|l_1-l_2|}^{l_1+l_2} V^{(l)}$$

The change-of-basis matrices are *Clebsch-Gordan (CG) coefficients* $C^{l_1,l_2,l}_{m_1,m_2,m}$. In equivariant networks, the tensor product layer:

$$(\mathbf{f}^{(l_1)} \otimes \mathbf{f}^{(l_2)})^{(l)}_m = \sum_{m_1,m_2} C^{l_1,l_2,l}_{m_1,m_2,m}\, f^{(l_1)}_{m_1}\, f^{(l_2)}_{m_2}$$

produces a degree-l output feature from two input features of degrees l_1 and l_2. Used in: e3nn (Geiger & Smidt, 2022), NequIP, MACE.

Use in equivariant networks. Node features in SE(3)/E(3)-equivariant networks are vectors of spherical harmonic coefficients, one per degree l. Convolution in the angular domain corresponds to multiplication by Wigner D-matrices; the CG decomposition enables mixing features of different degrees while preserving equivariance. The computational cost scales as $O(l_{\text{max}}^6)$ per CG contraction, which motivates truncation at small l_{max} (typically $l_{\text{max}} = 2$ or 3 for molecular systems).

Example — SphericalCNN on climate data

Cohen et al. (2018) apply an $S^2 \to SO(3)$ spherical CNN to climate pattern recognition. Input: ERA5 reanalysis data on the sphere ($1.4° \times 1.4°$ grid, 16 atmospheric variables). First layer: spherical convolution produces a function on $SO(3)$ (3 Euler angles); subsequent layers: $SO(3)$ group convolution. Output: detect atmospheric rivers and tropical cyclones. Spherical CNN: 97.9% detection accuracy. Standard CNN on equirectangular projection: 95.2% (distortion at poles hurts). The spherical CNN is equivariant to rotations of the globe, so a storm detected over the Pacific is recognised equally well over the Atlantic.

29.1 The SE(3) Equivariance Problem

Il problema fondamentale. Una molecola è una nuvola di punti $\{(\mathbf{r}_i, z_i)\}_{i=1}^N$ con posizioni $\mathbf{r}_i \in \mathbb{R}^3$ e tipi atomici z_i. Due configurazioni sono equivalenti se differiscono per una trasformazione $SE(3)$: $\{(\mathbf{R}\mathbf{r}_i + \mathbf{t}, z_i)\}$ per $\mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3$.

Quantità fisiche. L'energia potenziale E è $E(3)$-invariante (scalare). Il momento di dipolo □ è $O(3)$-equivariante come vettore di tipo-1 (si trasforma come $\mathbf{R}$□). Il tensore dei quadrupoli è equivariante come tensore di rango-2. Apprendere queste quantità richiede architetture che rispettino queste simmetrie per garantire consistenza fisica.

Architetture invarianti vs equivarianti.

- *Invarianti*: SchNet, DimeNet. Usano solo distanze $\|r_i - r_j\|$ e angoli, invarianti per $SE(3)$. Non possono predire quantità vettoriali.
- *Equivarianti*: EGNN, SE(3)-Transformer, PaiNN, NequIP. Mantengono feature equiv-

arianti di tipo-ℓ (campi tensoriali).

Feature tensoriali irreducibili. Una feature equivariante di tipo ℓ si trasforma come $\mathbf{f}^{(\ell)} \to \mathbf{D}^{(\ell)}(\mathbf{R})\mathbf{f}^{(\ell)}$ dove $\mathbf{D}^{(\ell)}$ è la matrice di Wigner di ordine ℓ ($(2\ell+1)\times(2\ell+1)$). $\ell=0$: scalare, $\ell=1$: vettore, $\ell=2$: tensore rango-2, ecc.

Prodotto tensoriale di Clebsch-Gordan. Per combinare feature di tipo ℓ_1 e ℓ_2:

$$(\mathbf{f}^{(\ell_1)} \otimes_{CG} \mathbf{f}^{(\ell_2)})^{(\ell)} = \sum_{m_1,m_2} C^{(\ell,m)}_{(\ell_1,m_1)(\ell_2,m_2)} f^{(\ell_1)}_{m_1} f^{(\ell_2)}_{m_2}$$

dove C sono i coefficienti di Clebsch-Gordan e $|\ell_1-\ell_2| \le \ell \le \ell_1+\ell_2$. Questo è il meccanismo core di e3nn e NequIP.

In molecular modelling, each atom i has:

- Scalar feature $\mathbf{h}_i \in \mathbb{R}^d$ (atom type, charge, ...)
- 3D coordinate $\mathbf{x}_i \in \mathbb{R}^3$

We want the learned representation to be invariant to global translation and equivariant to global rotation (rotating the molecule should rotate the learned 3D features accordingly).

SchNet: distance-based invariant network. SchNet (Schütt et al., 2017) uses only pairwise distances as inputs:

$$\mathbf{h}_i^{(l+1)} = \sum_{j\in\mathcal{N}(i)} \mathbf{h}_j^{(l)} \odot W(\|\mathbf{x}_i - \mathbf{x}_j\|)$$

where $W(d)$ is a continuous filter network taking a scalar distance as input. This is strictly $SE(3)$-invariant but cannot predict vectorial quantities. It is the baseline against which equivariant networks are measured.

Why SE(3)-equivariance matters for force fields In molecular dynamics, the force on atom i is $\mathbf{F}_i = -\nabla_{\mathbf{r}_i} E$. For a neural potential energy E_θ, the force is $\mathbf{F}_i = -\nabla_{\mathbf{r}_i} E_\theta(\mathbf{r}_1, \dots, \mathbf{r}_n)$. If E_θ is not SE(3)-invariant, rotating the molecule produces different energies — unphysical. Equivariant networks guarantee $E_\theta(R\mathbf{r}) = E_\theta(\mathbf{r})$ for all $R \in O(3)$, and forces transform as $\mathbf{F}(R\mathbf{r}) = R\mathbf{F}(\mathbf{r})$ — automatically consistent with Newton's third law. This reduces training data requirements by $3\times$ compared to non-equivariant networks that must learn invariance empirically from augmented training data.

Example — EGNN for molecular dynamics

Satorras et al. (2021) test EGNN on the MD17 molecular dynamics benchmark (aspirin, 950 training configurations). Input: atom positions $\mathbf{r}_i \in \mathbb{R}^3$, atom types $\mathbf{h}_i \in \mathbb{R}^{16}$. EGNN message: $\mathbf{m}_{ij} = \phi_e(\mathbf{h}_i, \mathbf{h}_j, \|\mathbf{r}_i - \mathbf{r}_j\|^2, \mathbf{a}_{ij})$; coordinate update: $\mathbf{r}_i' = \mathbf{r}_i + \frac{1}{|\mathcal{N}(i)|}\sum_j (\mathbf{r}_i - \mathbf{r}_j)\phi_x(\mathbf{m}_{ij})$. The scalar ϕ_x is learned; the equivariance arises from the direction $(\mathbf{r}_i - \mathbf{r}_j)$ being $O(3)$-equivariant by construction. EGNN achieves energy MAE of 0.29 kcal/mol — competitive with DimeNet++ at $3\times$ lower compute, demonstrating that $SE(3)$-equivariance provides strong data efficiency even without angular features.

29.2 EGNN Propagation Rules

Satorras et al. (2021) propongono EGNN: equivarianza $E(n)$ con semplicità computazionale, senza armoniche sferiche o CG-products.

Intuizione chiave. Per ottenere equivarianza senza armoniche sferiche, si usano differenze di posizione $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ (vettori) e distanze $d_{ij} = \|\mathbf{r}_{ij}\|$ (scalari invarianti) come input. Le posizioni si aggiornano come somma pesata di vettori di differenza, garantendo equivarianza.

EGNN update rules

$$\begin{aligned}
\mathbf{m}_{ij} &= \phi_e\left(h_i^{(\ell)}, h_j^{(\ell)}, d_{ij}^2, a_{ij}\right) \\
\mathbf{r}_i^{(\ell+1)} &= \mathbf{r}_i^{(\ell)} + C\sum_{j\neq i}(\mathbf{r}_i^{(\ell)} - \mathbf{r}_j^{(\ell)})\,\phi_x(\mathbf{m}_{ij}) \\
\mathbf{m}_i &= \sum_{j\in\mathcal{N}(i)} \mathbf{m}_{ij} \\
h_i^{(\ell+1)} &= \phi_h\left(h_i^{(\ell)}, \mathbf{m}_i\right)
\end{aligned}$$

dove ϕ_e, ϕ_x, ϕ_h sono MLP, a_{ij} attributi dell'arco, $C = 1/(|\mathcal{N}(i)| - 1)$ normalizzazione.

Prova di $E(n)$-equivarianza. L'aggiornamento di $\mathbf{r}_i$ è una combinazione lineare di $(\mathbf{r}_i - \mathbf{r}_j)$. Sotto $\mathbf{r} \to \mathbf{Q}\mathbf{r} + \mathbf{t}$ (con $\mathbf{Q} \in O(n)$): $(\mathbf{r}_i - \mathbf{r}_j) \to \mathbf{Q}(\mathbf{r}_i - \mathbf{r}_j)$, $d_{ij}^2 = \|\mathbf{r}_i - \mathbf{r}_j\|^2$ invariante. Quindi $\mathbf{r}_i^{(\ell+1)} \to \mathbf{Q}\mathbf{r}_i^{(\ell+1)} + \mathbf{t}$. Le feature scalari h_i dipendono solo da scalari invarianti, sono quindi invarianti.

Limiti di EGNN. Usa solo distanze (feature scalari), non riesce a distinguere enantiomeri (specchio chirali con energie diverse). Per proprietà chirali serve equivarianza $SO(3)$ ma non $O(3)$ (si rompe la simmetria di riflessione), richiedendo feature di parità definita.

EGNN Layer (Satorras et al., 2021)

For each edge (i, j):

$$m_{ij} = \phi_e\left(\mathbf{h}_i^{(l)}, \mathbf{h}_j^{(l)}, \left\|\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\right\|^2, a_{ij}\right)$$

For each node i:

$$\mathbf{h}_i^{(l+1)} = \phi_h\left(\mathbf{h}_i^{(l)}, \sum_{j\neq i} m_{ij}\right)$$

$$\mathbf{x}_i^{(l+1)} = \mathbf{x}_i^{(l)} + C\sum_{j\neq i}\left(\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}\right)\phi_x(m_{ij})$$

Why $SE(3)$-equivariant?

Translation invariance of m_{ij}: The edge message uses $\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2$ (interatomic distance), which is translation-invariant.

Rotation equivariance of coordinate update: The coordinate update is a weighted sum of displacement vectors $(\mathbf{x}_i - \mathbf{x}_j)$. Rotating all coordinates by $\mathbf{R}$ rotates each displacement by $\mathbf{R}$, so the update transforms as $\mathbf{R}(\mathbf{x}_i - \mathbf{x}_j)$ — equivariant by construction.

Invariance of $\mathbf{h}_i$*:* The scalar features use only distances (invariant), so $\mathbf{h}_i$ is invariant to rigid motions.

Example — EGNN coordinate update for atom i

Atom i at $(0,0,0)$, neighbours: atom j at $(1,0,0)$, atom k at $(0,1,0)$.
Suppose $\phi_x(m_{ij}) = 0.3$, $\phi_x(m_{ik}) = 0.5$:
$\Delta\mathbf{x}_i = C\left[0.3 \cdot (0,0,0) - (1,0,0)) + 0.5 \cdot ((0,0,0) - (0,1,0))\right] = C[-0.3\hat{e}_x - 0.5\hat{e}_y]$
Atom i moves away from both neighbours, weighted by message strength.

NequIP: Neural Equivariant Interatomic Potentials. Batzner et al. (2022) extend EGNN by incorporating full tensor feature representations (not just scalar h_i and vector coordinates). Each node carries features $\mathbf{f}_i^{(l)} = \bigoplus_\ell \mathbf{f}_i^{(l,\ell)}$ decomposed into irreps. The interaction layer uses Clebsch-Gordan products to mix angular momentum channels:

$$\mathbf{f}_i^{(l+1,\ell)} = \sum_j \sum_{\ell_1,\ell_2} C_{\ell_1\ell_2}^{\ell} \cdot \left(\mathbf{f}_j^{(l,\ell_1)} \otimes Y_{\ell_2}(\hat{r}_{ij})\right)^{(\ell)} R_{\ell_1\ell_2}^{\ell}(r_{ij})$$

where $Y_{\ell_2}(\hat{r}_{ij})$ are spherical harmonics of the unit displacement vector and R are learnable radial functions. NequIP achieves DFT-level accuracy on molecular dynamics with $100\times$ fewer training samples than previous methods.

PART 30

Optimal Transport for Deep Learning

The Monge-Kantorovich problem and Wasserstein distances provide a geometrically meaningful way to compare distributions. We derive the duality theorem and the Sinkhorn algorithm.

30.1 The Monge-Kantorovich Problem

Il *trasporto ottimo* (OT) fornisce una metrica geometrica tra distribuzioni di probabilità, fondamentale per loss functions nei modelli generativi, confronto di nuvole di punti, e allineamento di distribuzioni.

Problema di Monge (1781). Dati due spazi di probabilità $(\mathcal{X}, \mu)$ e $(\mathcal{Y}, \nu)$ con costo $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$:

$$\inf_{T : T_{\#}\mu = \nu} \int_{\mathcal{X}} c(x, T(x))\, d\mu(x)$$

dove $T_{\#}\mu$ è la misura push-forward. Il problema ha soluzioni non esistenti se μ è discreta.

Problema di Kantorovich (1942) — rilassamento.

$$W_p^p(\mu, \nu) = \inf_{\gamma \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} \|x - y\|^p \, d\gamma(x, y)$$

dove $\Pi(\mu, \nu) = \{\gamma \in \mathcal{P}(\mathcal{X} \times \mathcal{Y}) : \pi^1_{\#}\gamma = \mu,\ \pi^2_{\#}\gamma = \nu\}$ è l'insieme dei *piani di trasporto* (couplings). Sempre ha soluzione.

Distanza di Wasserstein

Per $p \geq 1$: $W_p(\mu, \nu) = \left(\inf_{\gamma \in \Pi(\mu,\nu)} \mathbb{E}_{(X,Y)\sim\gamma}[\|X - Y\|^p]\right)^{1/p}$. Per $p = 2$: distanza geodetica sullo spazio delle probabilità (geometria di Fisher-Rao).

Dualità di Kantorovich-Rubinstein. Per W_1:

$$W_1(\mu, \nu) = \sup_{\|f\|_L \leq 1} \left[\int f \, d\mu - \int f \, d\nu\right]$$

dove il sup è sulle funzioni 1-Lipschitz. Questa dualità è la base del WGAN: il discriminatore approssima la funzione 1-Lipschitz ottimale.

Definition: Optimal Transport

Given distributions μ and ν on $\mathbb{R}^d$, and a cost function $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$:

$$\mathrm{OT}(\mu, \nu) = \inf_{\gamma \in \Pi(\mu,\nu)} \int c(\mathbf{x}, \mathbf{y})\, d\gamma(\mathbf{x}, \mathbf{y})$$

where $\Pi(\mu, \nu)$ is the set of *transport plans* γ with marginals μ and ν.

For $c(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^p$, the p-th root gives the Wasserstein-p distance W_p.

Kantorovich duality.

$$\mathrm{OT}(\mu, \nu) = \sup_{f \oplus g \leq c} \left(\int f \, d\mu + \int g \, d\nu\right)$$

where $(f \oplus g)(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{y})$ and the constraint is $f(\mathbf{x}) + g(\mathbf{y}) \leq c(\mathbf{x}, \mathbf{y})$. For W_1 this specialises to the Kantorovich-Rubinstein duality.

Brenier's theorem. For $c(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$ and μ absolutely continuous, the optimal transport plan is deterministic: $\gamma^* = (\mathrm{Id}, T^*)_{\#}\mu$ where $T^* = \nabla\phi^*$ is the gradient of a convex function ϕ^* (the Brenier map). This means the optimal W_2 transport is a monotone rearrangement. In 1D this reduces to the quantile-matching map: $T^*(x) = F_\nu^{-1}(F_\mu(x))$.

Existence of Optimal Transport Plans The Kantorovich problem is a linear programme over the compact convex set $\Pi(\mu, \nu)$ of couplings. Since the objective $\langle \mathbf{C}, \mathbf{P} \rangle$ is continuous and $\Pi(\mu, \nu)$ is weakly compact (by Prokhorov's theorem for tight measures), an optimal coupling always exists. The *Birkhoff-von Neumann theorem* implies that for discrete distributions with n support points each, extreme points of $\Pi(\mathbf{a}, \mathbf{b})$ are permutation matrices (when $\mathbf{a} = \mathbf{b} = \mathbf{1}/n$) — the optimal plan is therefore a sparse matrix with at most $2n - 1$ non-zero entries, recovered as a basic feasible solution of the LP.

30.2 Discrete Optimal Transport

For discrete distributions $\mu = \sum_i a_i \delta_{\mathbf{x}_i}$ and $\nu = \sum_j b_j \delta_{\mathbf{y}_j}$ with $\mathbf{a}, \mathbf{b} \geq 0$, $\sum a_i = \sum b_j = 1$, the Kantorovich problem becomes a finite-dimensional linear programme:

$$\mathrm{OT}(\mu, \nu) = \min_{\mathbf{P} \in U(\mathbf{a}, \mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle, \qquad U(\mathbf{a}, \mathbf{b}) = \{\mathbf{P} \geq 0 : \mathbf{P}\mathbf{1} = \mathbf{a},\ \mathbf{P}^\top \mathbf{1} = \mathbf{b}\}$$

where $\mathbf{C}_{ij} = c(\mathbf{x}_i, \mathbf{y}_j)$ is the ground cost matrix. With n support points each, this is an n^2-variable LP with $2n$ equality constraints, solvable in $O(n^3)$ by the Hungarian algorithm or network simplex.

Dual linear programme. The Lagrangian dual reads:

$$\max_{\mathbf{f}, \mathbf{g}} \mathbf{f}^\top \mathbf{a} + \mathbf{g}^\top \mathbf{b} \quad \text{s.t.} \quad f_i + g_j \leq C_{ij} \ \ \forall\, i, j$$

The dual variables $(\mathbf{f}, \mathbf{g})$ are *Kantorovich potentials*. At optimality (strong duality), $f_i + g_j = C_{ij}$ for all (i, j) with $P^*_{ij} > 0$, and $f_i + g_j < C_{ij}$ otherwise — the complementary-slackness condition. The optimal plan is supported on the *c-cyclically monotone* set $\{(x, y) : f(x) + g(y) = c(x, y)\}$.

Wasserstein-p distances. For cost $c(x, y) = \|x - y\|^p$, the optimal transport cost defines the p-Wasserstein distance:

$$W_p(\mu, \nu) = \left(\min_{\mathbf{P} \in U(\mathbf{a}, \mathbf{b})} \sum_{i,j} \|x_i - y_j\|^p P_{ij} \right)^{1/p}$$

W_1 (Earth Mover's Distance) is solved efficiently by min-cost flow; W_2 (used in most deep learning applications) has the Brenier map characterisation.

Brenier's Theorem (discrete version)

For cost $c(x, y) = \|x - y\|^2$ and μ with distinct support points, the optimal transport plan is *deterministic*: $P^*_{ij} = a_i \, \mathbf{1}[j = T(i)]$ for a unique map $T : \{x_i\} \to \{y_j\}$. This map is the gradient of a convex function: $T = \nabla\phi$ for some discrete convex ϕ (i.e. T is a *monotone* map). For n points, T can be found by solving the assignment problem.

Computational complexity.

- *Exact LP / network simplex*: $O(n^3 \log n)$, infeasible for $n > 10^4$.
- *Entropic regularisation (Sinkhorn)*: $O(n^2/\varepsilon^2)$ iterations of $O(n^2)$ matrix-vector products; GPU-parallelisable and practical for $n \sim 10^5$.
- *Multiscale / hierarchical methods*: solve a coarse OT problem and refine, reducing effective n by a factor of 4–8 per level.
- *Sliced* W_1: linear in n by projection to 1D (see §30.6).

Sample complexity. For μ, ν supported on $[0,1]^d$, the empirical OT cost $\hat{W}_p^p$ estimated from n samples converges to the population value at rate $n^{-2/d}$ (for $d > 4$, Fournier & Guillin 2015) — the curse of dimensionality. Entropic regularisation improves this to $n^{-1/2}$ at the cost of bias $O(\varepsilon \log n)$.

30.3 Sinkhorn: Entropic Regularisation

Il problema OT discreto è un programma lineare di dimensione n^2, con complessità $O(n^3 \log n)$. Cuturi (2013) propone la regolarizzazione entropica per renderlo differenziabile e scalabile.

OT con regolarizzazione entropica.

$$\mathbf{P}^\star_\varepsilon = \arg\min_{\mathbf{P} \in U(\mathbf{a},\mathbf{b})} \langle \mathbf{P}, \mathbf{C} \rangle - \varepsilon H(\mathbf{P})$$

dove $H(\mathbf{P}) = -\sum_{ij} P_{ij}(\log P_{ij} - 1)$ è l'entropia di $\mathbf{P}$ e $U(\mathbf{a},\mathbf{b}) = \{\mathbf{P} \geq 0 : \mathbf{P}\mathbf{1} = \mathbf{a}, \mathbf{P}^\top \mathbf{1} = \mathbf{b}\}$.

Forma della soluzione. Per le condizioni KKT:

$$P^\star_{ij} = u_i \exp(-C_{ij}/\varepsilon) v_j = \operatorname{diag}(\mathbf{u})\, \mathbf{K} \operatorname{diag}(\mathbf{v})$$

dove $\mathbf{K} = \exp(-\mathbf{C}/\varepsilon)$ (element-wise), $\mathbf{u}, \mathbf{v} \geq 0$ moltiplicatori.

Algoritmo di Sinkhorn

$$\mathbf{v}^{(t+1)} = \mathbf{b} / (\mathbf{K}^\top \mathbf{u}^{(t)})$$
$$\mathbf{u}^{(t+1)} = \mathbf{a} / (\mathbf{K}\, \mathbf{v}^{(t+1)})$$

Convergenza lineare con tasso $O(e^{-t/\varepsilon})$. In log-space: log-sum-exp stabile numericamente.

Complessità. $O(n^2/\varepsilon^2)$ per una precisione ε-relativa. Molto meglio di $O(n^3)$. Implementabile su GPU con operazioni matriciali batch.

Differenziabilità. $W_\varepsilon(\mathbf{a},\mathbf{b})$ è differenziabile rispetto a $\mathbf{C}, \mathbf{a}, \mathbf{b}$, con gradiente $\partial W_\varepsilon / \partial C_{ij} = P^\star_{ij}$. Usato in: OT-based matching loss, Point Cloud Alignment (ShapeNet), Sliced-Wasserstein generative models.

Add an entropy penalty $\varepsilon H(\mathbf{P}) = -\varepsilon \sum_{ij} P_{ij}(\log P_{ij} - 1)$:

$$\mathrm{OT}_\varepsilon(\mu,\nu) = \min_{\mathbf{P} \in U(\mathbf{a},\mathbf{b})} \langle \mathbf{C}, \mathbf{P} \rangle - \varepsilon H(\mathbf{P})$$

Solution structure. The regularised optimal plan has the form:

$$P^*_{ij} = u_i\, K_{ij}\, v_j, \qquad K_{ij} = e^{-C_{ij}/\varepsilon}$$

for scaling vectors $\mathbf{u}, \mathbf{v} > 0$.

Sinkhorn Algorithm

Initialise $\mathbf{v} = \mathbf{1}$. Iterate:

$$\mathbf{u} \leftarrow \mathbf{a}/(\mathbf{K}\mathbf{v}), \qquad \mathbf{v} \leftarrow \mathbf{b}/(\mathbf{K}^\top \mathbf{u})$$

Converges in $O(N^2/\varepsilon^2)$ iterations. Each step is a matrix-vector product: $O(N^2)$.

Log-domain Sinkhorn (numerically stable). Work with $\mathbf{f} = \varepsilon \log \mathbf{u}$, $\mathbf{g} = \varepsilon \log \mathbf{v}$:

$$f_i \leftarrow \varepsilon \log a_i - \varepsilon \log \sum_j e^{(g_j - C_{ij})/\varepsilon}$$

Example — Sinkhorn on 2×2 problem

$\mathbf{a} = (0.6, 0.4)$, $\mathbf{b} = (0.4, 0.6)$, $\mathbf{C} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $\varepsilon = 0.5$.

$\mathbf{K} = e^{-\mathbf{C}/0.5} = \begin{pmatrix} 1 & e^{-2} \\ e^{-2} & 1 \end{pmatrix} \approx \begin{pmatrix} 1 & 0.135 \\ 0.135 & 1 \end{pmatrix}$

Iteration 1: $\mathbf{u} = (0.6, 0.4)/\mathbf{K}(1,1)^\top = (0.6/1.135,\ 0.4/1.135) \approx (0.529, 0.352)$

$\mathbf{v} = (0.4, 0.6)/\mathbf{K}^\top \mathbf{u} \approx (0.4/0.576,\ 0.6/0.530) \approx (0.694, 1.132)$

After $\sim$5 iterations, $\mathbf{P}^*$ concentrates mass on the diagonal (transport (x_1, y_1) and (x_2, y_2)), matching the intuition that $\mathbf{C}$ makes self-pairing free.

Example — Sinkhorn convergence rate

For cost matrix $\mathbf{C}$ with entries in $[0, 1]$ and regularisation $\varepsilon > 0$, the Sinkhorn algorithm converges to the entropic OT solution $\mathbf{P}^*_\varepsilon$ at linear rate: after k iterations, $\|\mathbf{P}_k - \mathbf{P}^*_\varepsilon\|_1 \le 2n^2 \exp(-k/(4n^2/\varepsilon))$ (Franklin & Lorenz, 1989). The number of iterations to reach error δ is $O(n^2 \log(n/\delta)/\varepsilon)$. In practice, using log-domain stabilisation (replacing $u = \exp(f/\varepsilon)$, $v = \exp(g/\varepsilon)$ with direct log-sum-exp updates) prevents numerical underflow for small ε, extending practical applicability to $\varepsilon \sim 10^{-3}$ without overflow.

30.4 Wasserstein Barycenter

The Wasserstein barycenter of distributions $\{p_k\}_{k=1}^K$ with weights $\{\lambda_k\}$ is the distribution p^* minimising the weighted Fréchet mean in Wasserstein space:

$$p^* = \arg\min_{p \in \mathcal{P}_2} \sum_{k=1}^K \lambda_k \, W_2^2(p, p_k)$$

Unlike the arithmetic mixture $\sum_k \lambda_k p_k$, the barycenter *interpolates geometry*: it blends shapes rather than averaging probability masses, avoiding blurring across modes.

Existence and uniqueness. When one of the p_k is absolutely continuous, the barycenter exists and is unique (Agueh & Carlier, 2011). It is characterised by the fixed-point condition: the optimal maps $T_k : p^* \to p_k$ satisfy $\sum_k \lambda_k T_k = \mathrm{id}$.

Fixed-point algorithm for discrete barycenters. Given K discrete distributions on N support points, iterate:

$$p^{(t+1)} \propto \exp\left(\sum_{k=1}^K \lambda_k \log T_k^{\#} \, p^{(t)} \right)$$

where T_k^* is the Brenier map from $p^{(t)}$ to p_k. With entropic regularisation this reduces to iterated Sinkhorn barycenter updates (Benamou et al., 2015): at each step, update K scaling vectors simultaneously until convergence.

Sinkhorn Barycenter Algorithm

Initialise $p^{(0)} = \text{Uniform}$. At iteration t:

1. For each k: run Sinkhorn between $p^{(t)}$ and p_k to get optimal plan $\mathbf{P}_k$.
2. Update: $p_i^{(t+1)} \propto \prod_{k=1}^{K} \left(\sum_j \frac{P_{k,ij}}{p_j^{(t)}} \right)^{\lambda_k} p_i^{(t)}$

Converges to the entropic regularised barycenter p_ε^*.

Free-support barycenters. If the support of p^* is not fixed, one alternates between: (i) solving for the optimal transport plan given the current support, and (ii) updating support positions via the weighted average of transported points. This is the *stochastic* Wasserstein barycenter algorithm of Claici et al. (2018), scalable to millions of points.

Applications.

- *Style transfer and texture synthesis*: barycenter of colour histograms blends two artistic styles in the W_2 sense.
- *Multi-domain adaptation*: given labelled source domains $\{p_k\}$, train on the barycenter rather than the mixture to avoid mode misalignment.
- *Data augmentation*: interpolate between training examples to create novel in-distribution samples (Wasserstein mixup).
- *Federated learning*: aggregate client model distributions via the Wasserstein barycenter instead of parameter averaging, preserving the geometry of the model space.

Example — Wasserstein barycenter for style transfer

Three artistic styles: Impressionism (colour histogram p_1), Cubism (p_2), Art Nouveau (p_3). Compute the W_2 barycenter with equal weights $\lambda_k = 1/3$ using the Sinkhorn barycenter algorithm (50 iterations). Result: a colour histogram p^* that blends the three palettes geometrically — warm ochres of Impressionism, muted greys of Cubism, and the organic greens of Art Nouveau, without the blurring of the arithmetic mixture $\frac{1}{3}(p_1 + p_2 + p_3)$. The barycenter can be recoloured onto any photograph by applying the optimal transport map $T : \mathbf{x} \mapsto \nabla\phi(\mathbf{x})$ where ϕ is the Brenier potential from the image's colour distribution to p^*.

30.5 Gromov-Wasserstein for Cross-Domain Comparison

Standard OT requires both distributions to live in the same metric space. **Gromov-Wasserstein (GW)** distance removes this requirement by comparing *intra-domain* pairwise distances, enabling transport between spaces of different dimensions and structure.

Formal definition. Let (X, d_1, μ) and (Y, d_2, ν) be two metric measure spaces. The GW distance is:

$$\mathrm{GW}_2^2(\mu, \nu) = \min_{\gamma \in \Pi(\mu,\nu)} \iint \left(d_1(x, x') - d_2(y, y') \right)^2 d\gamma(x, y)\, d\gamma(x', y')$$

The coupling γ matches points so that distances within X are as consistent as possible with distances within Y. Unlike standard OT, this is a *non-convex quartic* problem in γ.

Discrete formulation. For graphs with cost matrices $\mathbf{C}^X \in \mathbb{R}^{m\times m}$ and $\mathbf{C}^Y \in \mathbb{R}^{n\times n}$, node weight vectors $\mathbf{a} \in \Delta^m$, $\mathbf{b} \in \Delta^n$:

$$\mathrm{GW}(\mathbf{C}^X, \mathbf{C}^Y, \mathbf{a}, \mathbf{b}) = \min_{\mathbf{P}\in U(\mathbf{a},\mathbf{b})} \sum_{i,j,k,l} (C^X_{ik} - C^Y_{jl})^2 \, P_{ij} \, P_{kl}$$

This rewrites as $\langle \mathbf{P}, \mathbf{L}(\mathbf{C}^X, \mathbf{C}^Y, \mathbf{P})\rangle$ where $\mathbf{L}$ is a linearised cost matrix, enabling a *conditional gradient* (Frank-Wolfe) algorithm: each iteration solves a standard OT sub-problem with updated cost $\mathbf{L}(\mathbf{C}^X, \mathbf{C}^Y, \mathbf{P}^{(t)})$.

Entropic Gromov-Wasserstein. Adding entropic regularisation $-\varepsilon H(\mathbf{P})$:

$$\mathrm{GW}_\varepsilon = \min_{\mathbf{P}\in U(\mathbf{a},\mathbf{b})} \langle \mathbf{P}, \mathbf{L}(\mathbf{C}^X, \mathbf{C}^Y, \mathbf{P})\rangle - \varepsilon H(\mathbf{P})$$

Each conditional-gradient iteration now solves an entropically-regularised OT sub-problem via Sinkhorn, giving a practical $O(n^2)$-per-iteration scheme.

Fused Gromov-Wasserstein (FGW)

When nodes carry feature vectors $\mathbf{x}_i \in \mathbb{R}^d$ alongside graph structure, **Fused GW** interpolates between feature transport and structural transport:

$$\mathrm{FGW}_\alpha = \min_{\gamma\in\Pi(\mu,\nu)} \alpha\,\mathrm{GW}(\gamma) + (1-\alpha)\,\mathrm{OT}(\gamma), \qquad \alpha \in [0,1]$$

FGW is a proper metric on attributed graphs and is used for graph classification without pre-defined node correspondences.

Applications. GW applies wherever distributions live in incompatible spaces: graph comparison (molecules of different sizes), single-cell RNA-seq batch correction (aligning gene-expression manifolds across protocols), 3D shape correspondence in point clouds, and cross-lingual word-embedding alignment.

Example — GW for molecular graph comparison

Given molecules G_1 ($m = 12$ atoms) and G_2 ($n = 15$ atoms), set $C^X_{ik} =$ shortest-path(i,k) in G_1, $C^Y_{jl} =$ shortest-path(j,l) in G_2, uniform weights $a_i = 1/m$, $b_j = 1/n$. Running entropic GW ($\varepsilon = 0.01$) yields a coupling $\mathbf{P}^* \in \mathbb{R}^{12\times 15}$ aligning structurally similar atoms; the scalar GW_ε is a differentiable graph-similarity measure usable as a training loss.

GW distance and graph kernels The Gromov-Wasserstein distance between two graphs defines a *pseudo-metric* on the space of metric measure spaces (the GW distance is zero iff the two spaces are measure-preserving isometric). This enables using GW as a *graph kernel*: $k(G_1, G_2) = \exp(-\mathrm{GW}(G_1, G_2)/\sigma^2)$. Compared to the Weisfeiler-Lehman subtree kernel (which only compares local neighbourhoods), the GW kernel captures global structural similarity — two graphs can have identical local neighbourhoods but very different GW distances if their global distance distributions differ. Application: drug discovery, where two molecules with similar local chemistry but different ring systems must be distinguished.

30.6 Sliced Wasserstein Distance

Computing $W_p^p(\mu,\nu)$ in $\mathbb{R}^d$ requires solving an n^2-variable LP (cost $O(n^3)$ for $d>1$). Sliced Wasserstein exploits the closed-form solution in 1D to approximate W_p efficiently.

Definition. For a direction $\theta \in S^{d-1}$, let $\theta_{\#}\mu$ denote the pushforward of μ under the projection $\mathbf{x} \mapsto \theta \cdot \mathbf{x}$. The Sliced Wasserstein distance is:

$$\mathrm{SW}_p^p(\mu,\nu) = \int_{S^{d-1}} W_p^p(\theta_{\#}\mu,\, \theta_{\#}\nu)\, \sigma(d\theta)$$

where σ is the uniform measure on S^{d-1}. In 1D, $W_p(\alpha,\beta) = \|F_\alpha^{-1} - F_\beta^{-1}\|_{L^p}$ (the L^p distance between quantile functions), computable in $O(n\log n)$ after sorting.

Monte Carlo approximation. Sample L random directions $\theta_1,\dots,\theta_L \sim \mathrm{Uniform}(S^{d-1})$:

$$\widehat{\mathrm{SW}}_p^p = \frac{1}{L}\sum_{l=1}^{L} W_p^p(\theta_{l\#}\mu,\, \theta_{l\#}\nu)$$

Total cost: $O(Ln\log n)$, independent of d. For generative modelling, $L = 50$–200 projections suffice.

Max-Sliced Wasserstein

Instead of averaging, maximise over directions:

$$\mathrm{MaxSW}(\mu,\nu) = \max_{\theta\in S^{d-1}} W_p(\theta_{\#}\mu,\, \theta_{\#}\nu)$$

This provides a tighter lower bound on W_p and can be computed by projected gradient ascent on θ. Particularly useful when distributions differ in only a few directions (e.g. GANs where the generator learns most directions quickly but collapses in one).

Topological properties. SW_p metrises weak convergence on $\mathcal{P}_p(\mathbb{R}^d)$, the same topology as W_p. Convergence $\mathrm{SW}_p(\mu_n,\mu)\to 0$ iff $\mu_n \Rightarrow \mu$ and $\int \|\mathbf{x}\|^p\, d\mu_n \to \int \|\mathbf{x}\|^p\, d\mu$. However, $\mathrm{SW}_p \le C_{d,p}\, W_p$ (with a dimensional constant), so SW_p can be much smaller than W_p in high dimensions.

Applications.

- *Sliced-Wasserstein Autoencoder* (*SWA*): minimise SW_2 between the aggregated posterior $q(\mathbf{z})$ and the prior $p(\mathbf{z})$, avoiding the KL collapse of standard VAEs.
- *Sliced-Wasserstein GAN*: replace the Wasserstein critic with $\widehat{\mathrm{SW}}_2$; no Lipschitz constraint needed, more stable training.
- *Colour transfer*: match the colour histogram of a source image to a target using SW_2 iterative projection — provably converges to the W_2 optimal transfer.
- *Distributional robustness*: SW_2 balls are computationally tractable uncertainty sets for distributionally-robust optimisation.

Projected gradient ascent for MaxSW To compute $\mathrm{MaxSW}(\mu,\nu) = \max_{\theta\in S^{d-1}} W_1(\theta_{\#}\mu,\theta_{\#}\nu)$: initialise $\theta \sim \mathrm{Uniform}(S^{d-1})$, then iterate $\theta \leftarrow \Pi_{S^{d-1}}(\theta + \alpha\nabla_\theta W_1(\theta_{\#}\mu,\theta_{\#}\nu))$ where $\Pi_{S^{d-1}}$ is normalisation. The gradient $\nabla_\theta W_1(\theta_{\#}\mu,\theta_{\#}\nu) = \mathbb{E}_{x\sim\mu}[T^*(x\cdot\theta)\mathbf{x}] - \mathbb{E}_{y\sim\nu}[T^*(y\cdot\theta)\mathbf{y}]$ is computable in $O(n\log n)$

after sorting. Convergence: 20–50 gradient steps suffice for $d \leq 128$.

31.1 Contrastive Learning and InfoNCE

Il *contrastive learning* impara rappresentazioni confrontando coppie di esempi simili (positive) vs dissimili (negative), senza label supervisionate.

Framework generale. Data un'augmentazione $\mathcal{T}$, due view $\mathbf{x}_i = t(\mathbf{x})$ e $\mathbf{x}_j = t'(\mathbf{x})$ con $t, t' \sim \mathcal{T}$ sono la coppia positiva. Altre $N-1$ coppie dello stesso mini-batch sono negative.

InfoNCE Loss (van den Oord et al. 2018)

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{N} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

dove $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v}/(\|\mathbf{u}\|\|\mathbf{v}\|)$ (cosine similarity), $\tau > 0$ temperatura.

Interpretazione informazionale. Minimizzare $\mathcal{L}_{\text{InfoNCE}}$ massimizza un lower bound dell'informazione mutua $I(\mathbf{z}_i; \mathbf{z}_j)$:

$$I(\mathbf{z}_i; \mathbf{z}_j) \geq \log N - \mathcal{L}_{\text{InfoNCE}}$$

Con N negativi, si può stimare I fino a $\log N$ bit.

SimCLR (Chen et al. 2020). Encoder f_θ + projector g_ϕ (MLP 2-layer). La loss è calcolata sulle proiezioni $\mathbf{z}$, non sulle rappresentazioni $\mathbf{h} = f(\mathbf{x})$. Il projector si scarta al downstream task. Importanza critica di: (1) batch size grande ($N = 4096$), (2) augmentazioni forti (crop + color jitter + blur), (3) BN nel projector.

MoCo (He et al. 2020). Ovvia al bisogno di batch grandi con una *memory queue* di $K \sim 65536$ negativi e un encoder *momentum* f_k (aggiornato con EMA, non backprop): $\theta_k \leftarrow m\theta_k + (1-m)\theta_q$.

InfoNCE Loss

Given N image-text pairs $\{(\mathbf{v}_i, \mathbf{t}_i)\}_{i=1}^{N}$ with embeddings normalised to the unit sphere, and a temperature $\tau > 0$:

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp(\mathbf{v}_i \cdot \mathbf{t}_i/\tau)}{\sum_{j=1}^{N} \exp(\mathbf{v}_i \cdot \mathbf{t}_j/\tau)}$$

CLIP. Radford et al. (2021) train an image encoder f_θ and a text encoder g_ϕ jointly by minimising the symmetric InfoNCE loss:

$$\mathcal{L}_{\text{CLIP}} = \frac{1}{2}(\mathcal{L}_{\text{i2t}} + \mathcal{L}_{\text{t2i}})$$

where $\mathcal{L}_{\text{i2t}}$ is image-to-text (find matching text for each image) and $\mathcal{L}_{\text{t2i}}$ is the transpose.

Interpretation as mutual information. InfoNCE is a lower bound on the mutual information $I(\mathbf{V}; \mathbf{T})$:

$$I(\mathbf{V}; \mathbf{T}) \geq \log N - \mathcal{L}_{\text{InfoNCE}}$$

Minimising $\mathcal{L}_{\text{InfoNCE}}$ maximises a lower bound on the cross-modal mutual information.

Temperature τ. Small τ sharpens the softmax (hard contrastive), pushing negative pairs far apart but making training unstable. Large τ gives softer gradients. CLIP learns τ as a parameter (initialised to $\log(1/0.07)$).

Example — InfoNCE gradient for one positive pair

$N = 4$, $i = 1$, $s_{ij} = \mathbf{v}_1 \cdot \mathbf{t}_j/\tau$: suppose $s_{11} = 0.8$, $s_{12} = 0.1$, $s_{13} = 0.0$, $s_{14} = -0.1$.
Softmax: $Z = e^{0.8} + e^{0.1} + e^{0} + e^{-0.1} = 2.226 + 1.105 + 1.000 + 0.905 = 5.236$
$p_{11} = 2.226/5.236 = 0.425$. Loss for sample 1: $-\log(0.425) = 0.855$.
Gradient w.r.t. s_{11}: $(p_{11} - 1)/\tau = (0.425 - 1)/\tau$ — negative, correctly pulling $\mathbf{v}_1$ and $\mathbf{t}_1$ closer.

Uniformity and alignment. Wang & Isola (2020) decompose the contrastive objective into two terms:

$$\mathcal{L}_{\text{InfoNCE}} \approx \underbrace{-\mathbb{E}[\mathbf{z}_i \cdot \mathbf{z}_j^+]}_{\text{alignment}} + \underbrace{\log \mathbb{E}[e^{\mathbf{z}_i \cdot \mathbf{z}_k/\tau}]}_{\text{uniformity}}$$

Alignment pulls positive pairs together; *uniformity* pushes all embeddings to be spread uniformly on the hypersphere. Good representations require both: aligned (semantically similar things are nearby) and uniform (maximum information encoded, no wasted capacity). This geometric view on the unit sphere motivates the normalisation step in InfoNCE.

31.2 Cross-Attention for Vision-Language Fusion

Vision-language models must fuse information across two modalities with fundamentally different structure: images produce spatial feature maps $\mathbf{V} \in \mathbb{R}^{hw \times d_v}$ (patch tokens), while text produces sequential representations $\mathbf{T} \in \mathbb{R}^{n \times d_t}$ (word tokens). Cross-attention provides a flexible, learned mechanism for this fusion.

Cross-attention definition. Given a query sequence $\mathbf{Q}$ (from one modality) and key-value pairs $\mathbf{K}, \mathbf{V}$ (from another):

$$\text{CrossAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

Unlike self-attention (where $\mathbf{Q} = \mathbf{K} = \mathbf{V}$ come from the same sequence), cross-attention queries from one modality attend to keys and values from the other. This is the primary mechanism in encoder-decoder Transformers (the decoder's second attention block attends to the encoder output).

Bidirectional fusion. Most vision-language architectures use cross-attention in both directions:

- *Image-to-text*: text tokens $\mathbf{Q}_T = \mathbf{T}\mathbf{W}^Q$ attend to image patch keys/values $\mathbf{K}_V = \mathbf{V}\mathbf{W}^K$, $\mathbf{M}_V = \mathbf{V}\mathbf{W}^V$. Each text token gathers spatially-relevant visual information.
- *Text-to-image*: image patch tokens attend to text tokens for language-guided feature refinement (used in ViLBERT dual-stream, BLIP-2).

Gated Cross-Attention (Flamingo)

Alayrac et al. (2022) insert frozen *Perceiver Resampler* modules between the image encoder and LLM layers. The resampler compresses variable-size image features to

a fixed 64-token sequence via cross-attention (64 learned query tokens attend to all image patches). These 64 tokens are then injected into LLM layers via *gated cross-attention dense* (*GATED XATTN-DENSE*) layers:

$$\mathbf{y} = \mathbf{x} + \tanh(\alpha) \cdot \text{XATTN}(\mathbf{x}_{\text{text}}, \mathbf{x}_{\text{visual}})$$

where α is a learnable per-layer scalar initialised to 0. Initialising $\alpha = 0$ ensures that at the start of vision-language fine-tuning, the LLM output is unchanged — a form of residual initialisation that preserves language ability.

Q-Former (BLIP-2). Li et al. (2023) use a lightweight Querying Transformer (Q-Former) with $N = 32$ learnable query tokens. In the first stage, Q-Former is trained with three objectives (image-text contrastive, image-text matching, image-grounded text generation). In the second stage, the Q-Former output is projected to the LLM's embedding dimension and prepended to the text tokens, enabling frozen-LLM instruction following on vision tasks. This bootstraps vision-language capabilities at $\sim 1/100$th the compute of end-to-end training.

KV-cache in cross-attention At inference, the key-value pairs $\mathbf{K}_V = \mathbf{V}\mathbf{W}^K$, $\mathbf{M}_V = \mathbf{V}\mathbf{W}^V$ (the visual features) are fixed for a given image. They can be cached after the first forward pass, so each subsequent token generation step only requires computing the new query $\mathbf{q}_t\mathbf{W}^Q$ and the attention weights — an $O(m)$ operation rather than $O(nm)$. For Flamingo with $m = 64$ visual tokens and sequences of $n = 512$ text tokens, caching reduces cross-attention cost per token from $O(64 \cdot 512)$ to $O(64)$, a $512\times$ reduction in cross-attention FLOPs during autoregressive generation.

31.3 CLIP: Contrastive Language-Image Pre-Training

Radford et al. (2021) estendono il contrastive learning al setting multimodale immagine-testo, ottenendo rappresentazioni zero-shot con performance sorprendenti.

Architettura.

- Image encoder: ViT o ResNet, produce $\mathbf{v}_i \in \mathbb{R}^{d_v}$.
- Text encoder: Transformer causale o bidirezionale, produce $\mathbf{t}_i \in \mathbb{R}^{d_t}$.
- Proiezione: $\mathbf{z}_i^v = \mathbf{W}_v\mathbf{v}_i$, $\mathbf{z}_i^t = \mathbf{W}_t\mathbf{t}_i$, normalizzati a norma 1.

Loss simmetrica. Per un batch di N coppie (immagine, testo):

$$\mathcal{L} = -\frac{1}{2N}\left[\sum_i \log \frac{e^{s_{ii}/\tau}}{\sum_j e^{s_{ij}/\tau}} + \sum_i \log \frac{e^{s_{ii}/\tau}}{\sum_j e^{s_{ji}/\tau}}\right]$$

dove $s_{ij} = \mathbf{z}_i^v \cdot \mathbf{z}_j^t$ è la matrice di similarità $N \times N$. La matrice target è l'identità.

Zero-shot classification. Per classificare in K classi: si costruiscono K prompt "A photo of a [class]", si calcolano le embedding $\{\mathbf{t}_k\}$, si assegna $\hat{y} = \arg\max_k \mathbf{z}^v \cdot \mathbf{t}_k$.

Scala. Addestrato su WebImageText (WIT): $400M$ coppie immagine-testo raccolte dal web. ViT-L/14 raggiunge 76.2% top-1 su ImageNet zero-shot, competitivo con ResNet-50 supervisionato.

Limitazioni. Difficoltà con: (1) composizionalità fine-grained ("un gatto piccolo insegue un cane grande"), (2) conteggi precisi, (3) testo in-image, (4) generalizzazione a distribuzioni molto fuori distribuzione rispetto al web.

CLIP (Radford et al. 2021) trains a vision encoder f_θ and text encoder g_ϕ by maximising agreement between matched image-text pairs:

InfoNCE loss. Given a batch of N (image, text) pairs, with embeddings $\{\mathbf{z}_i^I = f(\mathbf{x}_i^I)\}$ and $\{\mathbf{z}_i^T = g(\mathbf{x}_i^T)\}$ (unit normalised), the loss is:

$$\mathcal{L} = -\frac{1}{2N}\sum_{i=1}^{N}\left[\log\frac{\exp(s(\mathbf{z}_i^I,\mathbf{z}_i^T)/\tau)}{\sum_{j=1}^{N}\exp(s(\mathbf{z}_i^I,\mathbf{z}_j^T)/\tau)} + \log\frac{\exp(s(\mathbf{z}_i^T,\mathbf{z}_i^I)/\tau)}{\sum_{j=1}^{N}\exp(s(\mathbf{z}_j^I,\mathbf{z}_i^T)/\tau)}\right]$$

where $s(\mathbf{u},\mathbf{v}) = \mathbf{u}^\top\mathbf{v}$ and τ is a learnable temperature.

Zero-shot classification. To classify image $\mathbf{x}$ into class c: encode class name as text $\mathbf{z}_c^T = g(\text{"A photo of a \{c\}"})$; predict $c^* = \arg\max_c s(f(\mathbf{x}), \mathbf{z}_c^T)$.

Why InfoNCE works. It is a lower bound on mutual information: $\mathcal{L}_{\text{InfoNCE}} \geq I(I;T) - \log N$.

> CLIP and zero-shot transfer CLIP's zero-shot image classification works as follows: for ImageNet 1K classes, create text prompts "a photo of a {class}" for each class, compute text embeddings $\mathbf{t}_1, \ldots, \mathbf{t}_{1000}$, then classify a test image $\mathbf{i}$ as $\arg\max_k \cos(\mathbf{i}, \mathbf{t}_k)$. Zero-shot CLIP (ViT-L/14@336) achieves 76.2% top-1 accuracy — matching a fully supervised ResNet-50 trained on ImageNet, without ever seeing an ImageNet label. On distribution-shifted benchmarks (ImageNet-V2, ObjectNet), CLIP degrades less than supervised models (-11% vs. -15% gap), suggesting better OOD generalisation from the diversity of web-scraped training data.

31.4 Flamingo: Few-Shot Vision-Language Models

Flamingo (Alayrac et al., 2022) is a large vision-language model designed for *few-shot in-context learning*: given a small number of image-text example pairs in the context, generate a natural-language response to a new image query — without gradient updates.

Architecture. Flamingo combines three frozen pre-trained components with new trainable modules:

1. **Vision encoder**: NFNet-F6 (normaliser-free ResNet) producing a spatial feature map $\mathbf{V} \in \mathbb{R}^{m \times d_V}$ for each image.
2. **Perceiver Resampler**: a cross-attention module with $N_q = 64$ learnable query tokens that compress the variable-length visual features to a fixed $64 \times d_{\text{LM}}$ sequence, enabling variable-resolution inputs.
3. **Frozen LLM** (Chinchilla 70B): processes interleaved text and image tokens.
4. **Gated cross-attention dense (GATED XATTN-DENSE)** layers: inserted between frozen LLM layers; text tokens attend to the 64 visual tokens from the Perceiver, with a tanh-gated residual initialised to zero (§31.2).

Interleaved image-text training. Flamingo is trained on a mixture of: (1) ALIGN-style image-text pairs; (2) LTIP (long text + image pairs from web documents); (3) VTP (video-text pairs). Crucially, Flamingo processes *sequences* of images and text — the LLM's causal

attention mask ensures that text tokens can attend to all preceding images but not future ones, enabling few-shot prompting with multiple examples.

Few-Shot Prompting Protocol

At inference, construct a prompt: $[I_1, Q_1, A_1, I_2, Q_2, A_2, \ldots, I_k, Q_k]$ with k examples (image I_i, question Q_i, answer A_i) followed by the query image and question. Flamingo generates A_k autoregressively. Performance scales log-linearly with k: 4-shot Flamingo-80B matches or exceeds fine-tuned specialist models on VQAv2, COCO captioning, and VATEX video captioning.

In-context learning mechanics. The gated cross-attention layers attend to the *most recent* image in the context (the ⟨img⟩ token preceding the current text position), while the LLM's self-attention over text provides reasoning and pattern-matching across examples. This separation of visual grounding (cross-attention) and cross-example reasoning (LLM self-attention) is key to Flamingo's few-shot ability.

Successors. OpenFlamingo (Awadalla et al., 2023) is an open-source reproduction; IDEFICS (Laurençon et al., 2023) extends it with improved instruction following; InstructBLIP (Dai et al., 2023) applies instruction tuning to the Q-Former architecture, achieving strong zero-shot performance by aligning visual features to instruction-following LLMs.

Open-vocabulary object detection with MLLMs Flamingo and subsequent MLLMs enable open-vocabulary detection: prompt "Describe all objects in this image and their positions." The LLM generates grounded descriptions without task-specific fine-tuning. *Grounding DINO* (Liu et al., 2023) trains a detector jointly with CLIP by aligning DINO features with text embeddings, enabling detection of any object named in natural language. On COCO zero-shot (48.5 AP), Grounding DINO surpasses supervised YOLO-v5 (48.3 AP) — the first zero-shot detector to match supervised performance, demonstrating that CLIP-aligned features transfer detection capability without bounding-box annotation.

32.1 Volume Rendering

Le *Neural Radiance Fields* (NeRF, Mildenhall et al. 2020) rappresentano una scena 3D come una funzione implicita neurale, apprendendo la radianza e la densità a ogni punto dello spazio.

Rappresentazione. $F_\theta : (\mathbf{x}, \mathbf{d}) \to (c, \sigma)$ dove $\mathbf{x} \in \mathbb{R}^3$ è la posizione, $\mathbf{d} \in S^2$ la direzione di vista, $\mathbf{c} \in [0,1]^3$ il colore RGB, $\sigma \geq 0$ la densità di volume.

Equazione di rendering volumetrico. Il colore osservato lungo un raggio $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ tra t_n e t_f:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma(\mathbf{r}(t))\, \mathbf{c}(\mathbf{r}(t), \mathbf{d})\, dt$$

dove $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))\, ds\right)$ è la *trasmittanza accumulata* (probabilità che il raggio arrivi a t senza essere assorbito).

Discretizzazione. Campionando N punti stratificati $t_1 < \cdots < t_N$, con $\delta_i = t_{i+1} - t_i$:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i (1 - e^{-\sigma_i \delta_i}) \mathbf{c}_i, \qquad T_i = \exp\left(-\sum_{j<i} \sigma_j \delta_j\right)$$

Training. Loss di ricostruzione fotometrica: $\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \|\hat{C}(\mathbf{r}) - C(\mathbf{r})\|_2^2$ su tutti i raggi di tutte le immagini di training. NeRF usa 2 reti: *coarse* per campionamento grossolano, *fine* per campionamento importance-weighted basato sulla distribuzione di σ coarse.

Volume Rendering Equation

A ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ (origin $\mathbf{o}$, direction $\mathbf{d}$) passes through a volume with density $\sigma(\mathbf{r}(t))$ and colour $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$. The expected colour:

$$\hat{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\, \sigma(\mathbf{r}(t))\, \mathbf{c}(\mathbf{r}(t), \mathbf{d})\, dt$$

where the *transmittance*:

$$T(t) = \exp\left(- \int_{t_n}^{t} \sigma(\mathbf{r}(s))\, ds \right)$$

is the probability that the ray travels from t_n to t without hitting any particle.

Discrete approximation. Stratified sampling: divide $[t_n, t_f]$ into N bins, sample t_i from each. For segment i with length $\delta_i = t_{i+1} - t_i$:

$$\hat{C}(\mathbf{r}) \approx \sum_{i=1}^{N} T_i \left(1 - e^{-\sigma_i \delta_i}\right) \mathbf{c}_i$$

$$T_i = \exp\left(- \sum_{j=1}^{i-1} \sigma_j \delta_j \right)$$

This is differentiable with respect to σ_i and $\mathbf{c}_i$ — the NeRF network outputs these for every sampled point.

Hierarchical sampling. Uniform sampling wastes queries in empty space. NeRF uses a coarse network to estimate the density distribution, then resamples proportionally to $T_i(1 - e^{-\sigma_i\delta_i})$ (the "contribution weight"). This importance sampling concentrates samples near surfaces.

Example — Differentiability of volume rendering

The key insight enabling backpropagation through NeRF: every operation in the discrete rendering equation is differentiable. For σ_i (density at sample i):

$$\frac{\partial \hat{C}}{\partial \sigma_i} = T_i \delta_i e^{-\sigma_i\delta_i} \mathbf{c}_i - \sum_{k>i} T_k (1 - e^{-\sigma_k\delta_k}) \mathbf{c}_k \cdot \delta_i$$

The first term increases colour contribution at i; the second decreases the transmittance for all subsequent samples, reducing their contribution. This gradient correctly penalises placing density in the wrong location.

32.2 Positional Encoding for High-Frequency Detail

A naive MLP mapping 3D coordinates $\mathbf{x} \in \mathbb{R}^3$ directly to colour and density fails to represent high-frequency scene detail: by the *spectral bias* of neural networks (Rahaman et al., 2019),

MLPs with standard activations preferentially learn low-frequency functions. Encoding inputs with high-frequency sinusoids before feeding them to the MLP overcomes this.

Fourier feature encoding. Mildenhall et al. (NeRF, 2020) map each coordinate to a $2L$-dimensional vector:

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \dots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p))$$

Applied independently to each of the 3 spatial coordinates and 2 viewing-direction spherical coordinates. With $L = 10$ for position ($\mathbb{R}^3 \to \mathbb{R}^{60}$) and $L = 4$ for direction ($\mathbb{R}^2 \to \mathbb{R}^{24}$), the MLP can faithfully represent details at the scale of $1/2^{L-1}$ — sub-pixel for typical scene bounds.

Theoretical justification. Tancik et al. (2020) analyse Fourier feature networks through the neural tangent kernel (NTK): without encoding, the NTK $K(x, y)$ of an MLP decays rapidly with $\|x-y\|$, limiting the effective bandwidth. Fourier features $\square(\mathbf{x}) = [\cos(2\pi\mathbf{B}\mathbf{x}), \sin(2\pi\mathbf{B}\mathbf{x})]$ (with random $\mathbf{B} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$) shift the NTK to $K(\mathbf{x}, \mathbf{y}) = \exp(-2\pi^2\sigma^2\|\mathbf{x} - \mathbf{y}\|^2/2)$ — a stationary kernel whose bandwidth is controlled by σ.

Hash Encoding (Instant NGP)

Müller et al. (2022) replace the sinusoidal encoding with a multi-resolution hash table. For each resolution l: compute the 8 surrounding voxel corners, hash their integer coordinates to T table entries, trilinearly interpolate the F-dimensional features. Concatenate features across L resolutions. This gives $O(T \cdot L \cdot F)$ parameters (typically $\sim$ 10MB) and reduces training from hours (NeRF) to seconds, by exploiting GPU-parallelism on the hash lookups and gradient accumulation.

Alias-free encoding. Barron et al. (2021) show that the standard NeRF positional encoding conflates samples at different scales (a point sample does not distinguish nearby regions). Mip-NeRF replaces point samples with *conical frustums* and encodes them with an *integrated positional encoding* (IPE): $\mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\square,\square)}[\gamma(\mathbf{x})]$, computable in closed form as $(\sin(2^l\pi\square) \odot e^{-2^{2l-1}\pi^2\text{diag}(\square)}, \dots)$. This yields anti-aliased, scale-aware radiance field representations.

Example — 3D Gaussian Splatting vs. NeRF

Kerbl et al. (2023) replace the implicit MLP of NeRF with an explicit set of 3D Gaussians $\{(\square_k, \square_k, \alpha_k, \mathbf{c}_k)\}$ (position, covariance, opacity, colour). Rendering: project Gaussians to 2D, sort by depth, alpha-composite. Training: initialise from SfM point cloud; optimise Gaussian parameters by gradient descent on rendering loss; adaptively split/clone/prune Gaussians. Result: $100\times$ faster rendering (> 100 FPS vs. 0.1 FPS for NeRF), at equal or better quality. The positional encoding mathematics of §32.2 does not apply to Gaussian Splatting (no MLP), but the same quality metric (PSNR, SSIM, LPIPS) is used for comparison.

32.3 SIREN: Sinusoidal Activations for Implicit Representations

Implicit Neural Representations (INRs) model a signal $s : \mathbb{R}^d \to \mathbb{R}^c$ as a continuous function f_θ parameterised by a neural network. Standard MLPs with ReLU activations have poor spectral properties for this task: their output is piecewise linear, limiting the representable frequency content, and their derivatives are piecewise constant, making them unsuitable for applications requiring higher-order derivative supervision.

SIREN architecture (Sitzmann et al., 2020). Replace ReLU with sin:

$$\mathbf{x}_{i+1} = \sin(\omega_0 \mathbf{W}_i \mathbf{x}_i + \mathbf{b}_i)$$

with $\omega_0 > 0$ controlling the frequency bandwidth (typically $\omega_0 = 30$). Initialisation: $\mathbf{W}_0 \sim \text{Uniform}(-1/d_{\text{in}}, 1/d_{\text{in}})$ for the first layer; $\mathbf{W}_i \sim \text{Uniform}(-\sqrt{6/d}, \sqrt{6/d})$ for subsequent layers (derived to maintain the distribution of activations across layers).

Why sinusoidal activations. The derivative of a SIREN layer is itself a SIREN (with cosine $= \sin(\cdot + \pi/2)$): $\partial_{\mathbf{x}}[\sin(\omega_0 \mathbf{W}\mathbf{x})] = \omega_0 \mathbf{W} \cos(\omega_0 \mathbf{W}\mathbf{x})$, another sinusoidal network with the same architecture. This means SIREN derivatives at all orders are well-defined, smooth, and representable — enabling losses involving ∇f, $\nabla^2 f$, or $\partial_t f$.

Fitting PDEs with SIREN

For a PDE $\mathcal{F}(\mathbf{x}, f, \nabla f, \nabla^2 f) = 0$ on domain Ω with boundary condition $f = g$ on $\partial\Omega$, train:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}\in\Omega}[\|\mathcal{F}(\mathbf{x}, f_\theta, \nabla f_\theta, \nabla^2 f_\theta)\|^2] + \lambda\, \mathbb{E}_{\mathbf{x}\in\partial\Omega}[\|f_\theta(\mathbf{x}) - g(\mathbf{x})\|^2]$$

Sitzmann et al. demonstrate this for the Helmholtz, Eikonal ($\|\nabla f\| = 1$), and Poisson equations. ReLU networks fail at the Helmholtz equation (oscillatory solutions); SIREN represents them faithfully with as few as 5 layers of 256 neurons.

Applications.

- *Image representation*: SIREN fits a 1024×1024 image to PSNR > 50dB in < 60s, vs. PSNR ~ 35dB for ReLU-MLP of comparable size.
- *3D shape* (*signed distance functions*): $f_\theta(\mathbf{x}) = \text{SDF}(\mathbf{x})$; trained on surface points and normals via the Eikonal loss $\|\nabla f\| = 1$ + zero-set constraint.
- *Video*: $f_\theta(x, y, t) = \text{RGB}(x, y, t)$; SIREN compresses video more efficiently than ReLU networks at equivalent parameter count.
- *Neural fields for physics*: fitting solutions to wave equations, heat equations; differential equation loss formulation above.

Neural radiance caching Müller et al. (2021) propose Neural Radiance Caching for real-time path tracing: an online-trained MLP (with hash encoding) predicts the outgoing radiance at any surface point and direction, acting as a cache for expensive Monte Carlo path tracing. The MLP is updated in < 5ms per frame using the most-recent path tracing estimates, achieving 30FPS rendering quality comparable to 1000 samples-per-pixel offline rendering. The positional encoding (§32.2) is critical: without it, the MLP cannot represent high-frequency radiance variations at material boundaries and glossy surfaces.

33.1 Multiple Sequence Alignment and Evolutionary Attention

AlphaFold2 (Jumper et al. 2021) rivoluziona la predizione della struttura proteica usando informazione evolutiva (MSA) e geometria 3D esplicita.

Multiple Sequence Alignment (MSA). Date N_{seq} sequenze omologhe di lunghezza L, la MSA $\mathbf{M} \in \{0, \dots, 20\}^{N_{\text{seq}} \times L}$ codifica mutazioni evolutive. Il principio di *analisi di covarianza*: residui fisicamente vicini nello spazio 3D co-evolvono; la covarianza statistica nella MSA segnala contatti.

MSA embedding. $\mathbf{M}$ è proiettata in una rappresentazione a coppie $\mathbf{z}_{ij} \in \mathbb{R}^{c_z}$ (per ogni coppia di residui i, j) e a righe $\mathbf{m}_{si} \in \mathbb{R}^{c_m}$ (per sequenza s, residuo i).

Row-wise gated self-attention.

$$\text{Att}_{ij}^{s} = \text{softmax}_j\left(\frac{(\mathbf{W}_Q\mathbf{m}_{si})^\top(\mathbf{W}_K\mathbf{m}_{sj})}{\sqrt{d}} + \mathbf{b}_{ij}\right)$$

dove $\mathbf{b}_{ij}$ è un bias aggiuntivo derivato dalle feature a coppia $\mathbf{z}_{ij}$ (information flow da pair representation a MSA). Questo crea un meccanismo di *pair bias* che introduce informazione strutturale nella attention sulla sequenza.

Column-wise attention. Uguale ma lungo le colonne della MSA (allineando informazione dello stesso residuo attraverso sequenze diverse). Estrae coevolution signal senza calcolare direttamente la covarianza.

Outer product mean. Aggiorna la pair representation $\mathbf{z}_{ij}$ dalla MSA row:

$$\mathbf{z}_{ij} \mathrel{+}= \frac{1}{N_{\text{seq}}}\sum_s (\mathbf{W}_a\mathbf{m}_{si}) \otimes (\mathbf{W}_b\mathbf{m}_{sj})$$

Il prodotto esterno cattura la covarianza tra residuo i e residuo j attraverso tutte le sequenze.

A protein of L residues is represented by its sequence plus a multiple sequence alignment (MSA) of S homologous sequences. The MSA matrix $\mathbf{M} \in \{1, \dots, 21\}^{S\times L}$ (21 amino acid types + gap).

Row-wise attention (across residues for each sequence s):

$$\text{RowAttn}(\mathbf{M})_{s,i} = \text{softmax}\left(\frac{\mathbf{Q}_s\mathbf{K}_s^\top}{\sqrt{d_k}}\right)\mathbf{V}_s$$

Column-wise attention (across sequences for each position i):

$$\text{ColAttn}(\mathbf{M})_{s,i} = \text{softmax}\left(\frac{\mathbf{Q}_{\cdot,i}\mathbf{K}_{\cdot,i}^\top}{\sqrt{d_k}}\right)\mathbf{V}_{\cdot,i}$$

Column attention captures co-evolutionary signals: if position i tends to have residue A when position j has residue B across many homologs, this co-variation is a structural contact signal.

The Evoformer block. AlphaFold2's core module, the Evoformer, iterates between MSA stack (row/column attention) and pair stack (triangle updates). The triangle multiplicative update aggregates information from triangles of residues:

$$\mathbf{z}_{ij}^{\text{out}} = \text{LayerNorm}\left(\mathbf{z}_{ij} + \sum_k \mathbf{z}_{ik} \odot \mathbf{z}_{kj}\right)$$

This enforces a form of triangle inequality on pairwise distances: if i is close to k and k is close to j, then i is likely close to j. The 48 Evoformer blocks iteratively refine both the MSA representation and the pair representation before the structure module.

33.2 The Invariant Point Attention (IPA) Module

The AlphaFold2 structure module (Jumper et al., 2021) operates on per-residue frames $T_i = (\mathbf{R}_i, \mathbf{t}_i) \in SE(3)$. It must be *equivariant*: rotating the entire protein globally should rotate all predicted frames by the same rotation. IPA achieves this by computing distances between 3D points expressed in local frames — distances that are invariant to global rigid-body transformations.

Local frame points. For each head p and residue i, IPA projects the single-representation $\mathbf{s}_i \in \mathbb{R}^{c_s}$ to query/key/value *points* in the local frame:

$$\mathbf{q}_i^{(p)} = T_i \circ (\mathbf{W}_p^q \mathbf{s}_i), \quad \mathbf{k}_j^{(p)} = T_j \circ (\mathbf{W}_p^k \mathbf{s}_j), \quad \mathbf{v}_j^{(p)} = T_j \circ (\mathbf{W}_p^v \mathbf{s}_j)$$

where $T \circ \mathbf{x} \triangleq \mathbf{R}\mathbf{x} + \mathbf{t}$ maps a point from the local frame into the global frame.

Attention logit. IPA combines three additive terms:

$$a_{ij} = w_L \cdot \frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{c}} + w_C \cdot b_{ij}^{\text{pair}} - \frac{w_P}{2} \sum_p \left\| \mathbf{q}_i^{(p)} - \mathbf{k}_j^{(p)} \right\|^2$$

The first term is standard dot-product attention on the sequence representation. The second is a learned pair bias b_{ij} from the pair representation. The third penalises large inter-point distances in 3D — this term is $SE(3)$-invariant because $\|\mathbf{q}_i^{(p)} - \mathbf{k}_j^{(p)}\|$ depends only on relative frame geometry, not on global orientation.

SE(3)-Invariance of IPA Distance Term

Under a global rigid motion $g = (\mathbf{R}_g, \mathbf{t}_g)$: each frame transforms as $T_i \mapsto g \circ T_i$. The point $\mathbf{q}_i^{(p)}$ becomes $(g \circ T_i) \circ (\mathbf{W}^q \mathbf{s}_i) = \mathbf{R}_g \mathbf{q}_i^{(p)} + \mathbf{t}_g$. The *difference* $\mathbf{q}_i^{(p)} - \mathbf{k}_j^{(p)}$ picks up only the global rotation $\mathbf{R}_g$, and its Euclidean norm $\|\mathbf{R}_g(\mathbf{q}_i^{(p)} - \mathbf{k}_j^{(p)})\| = \|\mathbf{q}_i^{(p)} - \mathbf{k}_j^{(p)}\|$ is unchanged. Hence a_{ij} is invariant to global rigid motions.

Equivariant output aggregation. After soft-max, IPA aggregates both scalar and point-valued outputs:

$$\mathbf{o}_i = \text{concat}\left(\sum_j \alpha_{ij} \mathbf{v}_j, \ \sum_j \alpha_{ij} T_i^{-1} \circ \mathbf{v}_j^{(p)}, \ \sum_j \alpha_{ij} \mathbf{z}_{ij} \right)$$

The term $T_i^{-1} \circ \mathbf{v}_j^{(p)}$ expresses the aggregated value point in the *local* frame of residue i; this makes $\mathbf{o}_i$ equivariant: rotating the protein rotates $\mathbf{o}_i$ accordingly, so downstream frame updates remain consistent.

IPA in context. IPA is the attention mechanism inside each of the $N_s = 8$ structure-module layers. After IPA, a shared-parameter transition network updates $\mathbf{s}_i$, followed by the backbone-frame update (§33.3) and torsion-angle prediction. The iterative geometry-aware updates allow the module to refine structure from a completely unfolded state to near-crystallographic accuracy (< 1 Å RMSD on CASP14 free-modelling targets).

IPA across the 8 structure-module layers AlphaFold2 applies 8 iterations of IPA + backbone frame update, reusing the same weights across iterations (recycling). At iteration 1, frames are initialised as $T_i = (\mathbf{I}, \mathbf{0})$ — all residues at the origin. IPA uses the inter-point distances $\|\mathbf{q}_i^{(p)} - \mathbf{k}_j^{(p)}\|$, which are all zero initially (every frame maps to

the origin). As iterations proceed, the frame updates ΔT_i push frames into distinct positions, and the distance term in the attention logit becomes informative. The 8-layer depth is a deliberate design: too few layers and the geometry cannot converge; too many and the structure module overfits the recycling procedure.

33.3 Backbone Frame Update

After each IPA layer, the per-residue representations $\mathbf{s}_i$ are updated and used to predict a *frame update* $\Delta T_i \in SE(3)$ that refines the current backbone geometry.

Frame update via Gram-Schmidt. A small network maps $\mathbf{s}_i$ to two 3D vectors $(\mathbf{a}_i, \mathbf{b}_i) \in \mathbb{R}^3$, which are orthogonalised to form a rotation matrix:

$$\hat{e}_1 = \frac{\mathbf{a}_i}{\|\mathbf{a}_i\|}, \quad \hat{e}_2 = \frac{\mathbf{b}_i - (\hat{e}_1 \cdot \mathbf{b}_i)\hat{e}_1}{\|\cdots\|}, \quad \hat{e}_3 = \hat{e}_1 \times \hat{e}_2$$

$$\Delta \mathbf{R}_i = [\hat{e}_1\ \hat{e}_2\ \hat{e}_3] \in SO(3), \qquad \Delta \mathbf{t}_i = \mathbf{W}_t \mathbf{s}_i$$

The updated frame is $T_i' = T_i \circ \Delta T_i$ — the update is applied in the *local* frame of residue i, ensuring equivariance.

Why left-composition preserves equivariance. Under global rotation g: $T_i \mapsto gT_i$, and $(gT_i) \circ \Delta T_i = g(T_i \circ \Delta T_i)$. The update ΔT_i depends only on $\mathbf{s}_i$ (sequence/pair features, not absolute coordinates), so applying g post-hoc still yields the same result — the module is globally equivariant.

Torsion-angle prediction. Backbone frames fix C_α positions and orientations; full-atom coordinates require side-chain torsion angles $\chi_1, \ldots, \chi_4$. A separate MLP predicts the $(\sin \chi_k, \cos \chi_k)$ pairs, from which atomic positions are computed by recursive rotation around bond axes. AlphaFold2 predicts all 7 backbone and side-chain torsions jointly.

Structure module loss. Training uses a *Frame-Aligned Point Error* (*FAPE*) loss:

$$\mathcal{L}_{\text{FAPE}} = \frac{1}{|T||x|} \sum_{k,l} \left\| T_k^{-1} \circ \hat{\mathbf{x}}_l - T_k^{-1} \circ \mathbf{x}_l \right\|_\varepsilon$$

where $\hat{\mathbf{x}}_l$ are predicted atom positions, $\mathbf{x}_l$ are ground-truth positions, and $T_k^{-1} \circ (\cdot)$ expresses positions in each frame's local coordinate system. Clamping at $d_{\text{clamp}} = 10\,\text{Å}$ with Huber norm $\|\cdot\|_\varepsilon$ ensures robustness to rare gross errors. FAPE is the principal structural supervision signal and accounts for roughly 60% of the total AlphaFold2 training loss.

Example — AlphaFold2 on CASP14

At CASP14 (2020), AlphaFold2 achieved median TM-score 0.92 across 87 free-modelling targets — a 0.25 improvement over the next-best predictor (RoseTTAFold: 0.80). For the hardest targets (those with $< 30\%$ sequence identity to any known structure), AF2 achieved TM-score > 0.8 on 70% of targets, vs. $< 30\%$ for all prior methods. The structure module with 8 IPA layers (§33.2) + torsion-angle prediction + recycling (3 passes through Evoformer+structure module) accounts for the bulk of this performance. The FAPE loss (§33.3) was critical: prior work using RMSD as the training loss was less robust to outliers and inconsistent across different chain length normalisations.

33.4 Template Pair Stack and Structure Module

AlphaFold 2's **structure module** directly outputs 3D backbone frames from the pair representation $\mathbf{z}_{ij}$ and single representation $\mathbf{s}_i$.

Backbone frame. Each residue i is represented by a rigid frame $(R_i, \mathbf{t}_i) \in SE(3)$ (rotation + translation). The frame is updated by:

$$(R_i, \mathbf{t}_i) \leftarrow (R_i, \mathbf{t}_i) \circ \Delta(R_i, \mathbf{t}_i)$$

where Δ is predicted by a network from the updated single representation.

Invariant Point Attention (IPA). Attends over frames using both sequence features and 3D geometry:

$$a_{ij} \propto \exp\left(\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d}} - w \sum_p \| R_i \tilde{\mathbf{q}}_{ip} - R_j \tilde{\mathbf{k}}_{jp} \|^2 \right)$$

where $\tilde{\mathbf{q}}_{ip}, \tilde{\mathbf{k}}_{jp}$ are learned 3D query/key points in local frames. The $SE(3)$-invariant distance term allows the model to reason about 3D geometry without hard-coding it.

FAPE loss (Frame Aligned Point Error). Measures backbone accuracy invariant to rigid-body transformations:

$$\text{FAPE} = \frac{1}{N_{\text{frames}} N_{\text{atoms}}} \sum_{l,i} \left\| \hat{R}_l^{-1}(\hat{\mathbf{x}}_i - \hat{\mathbf{t}}_l) - R_l^{-1}(\mathbf{x}_i - \mathbf{t}_l) \right\|$$

By evaluating positions in local frames, FAPE is invariant to global rotation/translation and focuses on local structural accuracy.

Side-chain torsion prediction. Backbone frames fix C_α, N, and C positions. AlphaFold2 additionally predicts side-chain torsion angles $\chi_1, \ldots, \chi_4$ (up to 4 per residue) via a small torsion head. The torsion angles define the full all-atom geometry via forward kinematics: given the backbone frame, each successive bond is rotated by the corresponding χ angle to place the side-chain atoms.

PART 34

Manifold Learning and Topological Data Analysis

Real data often lies on a low-dimensional manifold in high-dimensional space. We formalise manifolds, differential geometry fundamentals, and persistent homology.

Evoformer: triangular update modules The Evoformer processes the pair representation $\mathbf{z}_{ij} \in \mathbb{R}^{c_z}$ with two types of triangular update. *Outgoing triangular update*: for each edge (i, j), aggregate information from all edges (i, k) and (j, k) sharing a common "apex" k: $\mathbf{z}_{ij} \mathrel{+}= \text{LayerNorm}\left(\sum_k a_k \, \mathbf{b}_k\right)$ where $a_k = \sigma(\mathbf{W}_a \mathbf{z}_{ik})$ and $\mathbf{b}_k = \mathbf{W}_b \mathbf{z}_{jk}$. This enforces triangle inequality-like constraints: if residue i is close to k and j is close to k, then i and j must be related. The *incoming* variant sums over k with roles reversed. Together, triangular updates propagate geometric constraints through the pair representation before the structure module uses it.

34.1 Smooth Manifolds

La *geometria differenziale* fornisce il linguaggio per descrivere lo spazio latente dei modelli generativi, le traiettorie nello spazio dei parametri, e le simmetrie continue.

Varietà differenziabile

Una varietà $\mathcal{M}$ di dimensione d è uno spazio topologico tale che ogni punto $p \in \mathcal{M}$ ha un intorno omeomorfo a $\mathbb{R}^d$ (carta locale). Se le mappe di transizione tra carte sono C^∞, la varietà è *liscia* (smooth).

Esempi.

- S^n: sfera n-dimensionale in $\mathbb{R}^{n+1}$.
- $SO(3)$: varietà di dim-3 (non omeomorfa a S^3 ma a $\mathbb{R}P^3$).
- Spazio latente VAE: tipicamente assunto essere $\mathbb{R}^d$ ma la varietà dei dati è immersa in esso.
- Spazio dei parametri di una rete: $\mathbb{R}^{|\theta|}$, ma la metrica informazionale ne fa una varietà Riemanniana (cfr. Vol. IV).

Spazio tangente e vettori tangenti. Lo spazio tangente $T_p\mathcal{M}$ in p è lo spazio vettoriale delle direzioni di curve lisce passanti per p. Dim$= d$. La struttura Riemanniana aggiunge una metrica $g_p : T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$ variante con il punto.

Geodetiche. Curve di lunghezza minima su $\mathcal{M}$ (generalizzano le linee rette). Soddisfano l'equazione differenziale delle geodetiche con simboli di Christoffel Γ^k_{ij}. L'interpolazione nello spazio latente lungo geodetiche (invece che linee rette euclidee) produce interpolazioni semanticamente più significative.

Curvatura. Il tensore di Riemann R^l_{kij} misura la non-commutatività del trasporto parallelo. La curvatura scalare $\kappa < 0$ indica spazio iperbolico (alberi, gerarchie); $\kappa > 0$ sfera. Gli spazi iperbolici sono utili per embedding di gerarchie (word embeddings iperbolici).

Definition: Smooth Manifold

A smooth d-dimensional manifold $\mathcal{M}$ is a topological space that locally resembles $\mathbb{R}^d$: for every point $p \in \mathcal{M}$, there exists a neighbourhood $U \ni p$ and a diffeomorphism (smooth bijection with smooth inverse) $\phi : U \to V \subseteq \mathbb{R}^d$.

Tangent space. At $p \in \mathcal{M}$, the tangent space $T_p\mathcal{M}$ is the vector space of all directions of smooth curves through p. It has dimension d.

Riemannian metric. A Riemannian metric assigns an inner product $g_p : T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$ that varies smoothly with p. The metric allows measuring lengths, angles, and distances on $\mathcal{M}$.

Manifold hypothesis. High-dimensional data (images, text embeddings) lies near a low-dimensional manifold $\mathcal{M} \subset \mathbb{R}^D$ with $d \ll D$. Neural networks learn functions on $\mathcal{M}$, not on all of $\mathbb{R}^D$.

Intrinsic vs ambient dimensionality. The intrinsic dimension d of a dataset can be estimated from the data alone using, e.g., the correlation dimension or the TwoNN estimator (Facco et al., 2017). For ImageNet embeddings from a ResNet, the intrinsic dimension is estimated at ≈ 40, far below the ambient dimension of 2048. This gap motivates dimensionality reduction and manifold-aware regularisation.

34.2 Geodesics and Exponential Map

On a smooth Riemannian manifold $(\mathcal{M}, g)$, a **geodesic** is a curve $\gamma : [0, 1] \to \mathcal{M}$ that locally minimises arc length, satisfying the geodesic equation:

$$\ddot{\gamma}^k + \Gamma^k_{ij} \dot{\gamma}^i \dot{\gamma}^j = 0$$

where $\Gamma^k_{ij} = \frac{1}{2} g^{kl}(\partial_i g_{jl} + \partial_j g_{il} - \partial_l g_{ij})$ are the Christoffel symbols. Geodesics generalise straight lines: they have zero covariant acceleration and are the "straightest" curves on the manifold.

Exponential map. The exponential map $\operatorname{Exp}_p : T_p\mathcal{M} \to \mathcal{M}$ maps a tangent vector $\mathbf{v} \in T_p\mathcal{M}$ to the endpoint of the unit-speed geodesic starting at p in direction $\mathbf{v}/\|\mathbf{v}\|$ at time $\|\mathbf{v}\|$:

$$\operatorname{Exp}_p(\mathbf{v}) = \gamma(1), \quad \gamma(0) = p,\ \dot{\gamma}(0) = \mathbf{v}$$

The logarithmic map $\operatorname{Log}_p : \mathcal{M} \to T_p\mathcal{M}$ is the local inverse: $\operatorname{Log}_p(q) = \mathbf{v}$ such that $\operatorname{Exp}_p(\mathbf{v}) = q$ and $\|\mathbf{v}\| = d(p, q)$.

Closed forms for key manifolds.

- *Sphere* S^n: $\operatorname{Exp}_p(\mathbf{v}) = \cos(\|\mathbf{v}\|)p + \sin(\|\mathbf{v}\|)\mathbf{v}/\|\mathbf{v}\|$; $\operatorname{Log}_p(q) = \frac{\theta}{\sin\theta}(q - \cos\theta\, p)$ with $\theta = \arccos(\langle p, q\rangle)$.
- *Poincaré ball* $\mathbb{B}^n_c$: $\operatorname{Exp}^c_{\mathbf{x}}(\mathbf{v}) = \mathbf{x} \oplus_c \left(\tanh(\frac{\sqrt{c}}{2}\lambda_{\mathbf{x}}\|\mathbf{v}\|)\frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|}\right)$ where $\lambda_{\mathbf{x}} = \frac{2}{1-c\|\mathbf{x}\|^2}$.
- *SPD matrices* Sym^+_n: $\operatorname{Exp}_P(S) = P^{1/2}\exp(P^{-1/2}SP^{-1/2})P^{1/2}$.
- *SO(3)*: $\operatorname{Exp}_I(\square) = \exp(\square)$ (matrix exponential for skew-symmetric $\square$; Rodrigues formula).

Parallel Transport

Parallel transport $\mathcal{P}_{p\to q} : T_p\mathcal{M} \to T_q\mathcal{M}$ moves tangent vectors along a geodesic without rotating them relative to the manifold geometry. Gradient descent on $\mathcal{M}$ requires transporting momentum/velocity vectors along the update trajectory (used in Riemannian Adam). On the Poincaré ball, parallel transport has a closed form via the Möbius gyration operator.

Cut locus. The exponential map is a diffeomorphism only within the *injectivity radius* $r_{\text{inj}}(p)$ — the largest ball around p on which Exp_p is injective. Beyond the cut locus $C(p) = \text{Exp}_p(\partial B(0, r_{\text{inj}}))$, Log_p is not well-defined, and geodesics may fail to be globally minimising. For optimisation purposes, initialising iterates inside $B(p, r_{\text{inj}})$ and using small step sizes avoids this singularity.

Example — Exponential map on $SO(3)$

For a rotation in 3D: any rotation $\mathbf{R} \in SO(3)$ near the identity can be written as $\mathbf{R} = \exp(\square)$ where $\square$ is skew-symmetric ($\square^\top = -\square$). The Rodrigues formula gives the closed form: $\exp(\square) = \mathbf{I} + \frac{\sin\theta}{\theta}\square + \frac{1-\cos\theta}{\theta^2}\square^2$ with $\theta = \|\square\|_F/\sqrt{2}$ (the rotation angle). The log map: $\text{Log}(\mathbf{R}) = \frac{\theta}{2\sin\theta}(\mathbf{R} - \mathbf{R}^\top)$ for $\theta \in (0, \pi)$. At $\theta = \pi$ (180° rotation), the log map is not unique — the cut locus of $SO(3)$ at the identity is the set of 180° rotations, which forms a 2-sphere in $\mathbb{R}^3$.

34.3 Persistent Homology

Persistent homology is a tool from algebraic topology that quantifies multi-scale topological features (connected components, loops, voids) of a dataset and summarises them in a compact, stable descriptor.

Vietoris-Rips filtration. Given a point cloud $\mathcal{X} \subset \mathbb{R}^d$ and a scale parameter $\epsilon \geq 0$, the Vietoris-Rips complex $\text{VR}(\mathcal{X}, \epsilon)$ includes a k-simplex for every $(k+1)$-subset of $\mathcal{X}$ with pairwise distances $\leq \epsilon$. As ϵ increases from 0 to ∞, simplices are added monotonically, forming a *filtration* $\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_N$.

Persistent homology. For each dimension p (0 = components, 1 = loops, 2 = voids), track when a p-dimensional homological feature is *born* (at scale ϵ_b) and *dies* (at scale $\epsilon_d > \epsilon_b$). The collection of birth-death pairs:

$$\text{dgm}_p(\mathcal{X}) = \{(\epsilon_b^i, \epsilon_d^i)\}_i \subset \mathbb{R}^2_<$$

is the *persistence diagram*. Features with large *persistence* $\epsilon_d - \epsilon_b$ are topologically significant; those near the diagonal $\epsilon_d \approx \epsilon_b$ are noise.

Stability Theorem (Cohen-Steiner et al., 2007)

The bottleneck distance between persistence diagrams is stable under perturbations:

$$d_\infty(\text{dgm}(\mathcal{X}), \text{dgm}(\mathcal{Y})) \leq \|\mathcal{X} - \mathcal{Y}\|_\infty$$

where d_∞ is the bottleneck distance (optimal matching between diagrams, measuring max cost). This guarantees that small noise in the data produces small changes in the topological descriptor.

Persistence images and vectorisation. Raw persistence diagrams are not amenable to gradient-based learning. Common vectorisations:

- *Persistence landscape*: $\lambda_k(\epsilon) = k$-th largest value of $\{\min(\epsilon - \epsilon_b^i, \epsilon_d^i - \epsilon)\}_i$; lives in L^2, permitting averaging.
- *Persistence image*: convolve the diagram with a Gaussian kernel on the half-plane to obtain a finite-dimensional vector; differentiable with respect to input point positions.
- *Topological signature*: statistics (total persistence, Betti numbers at fixed scale) as features for downstream classifiers.

TDA in machine learning. Topological features complement geometric and spectral features: they capture global shape (e.g. number of holes in a molecule) that local neighbourhood aggregation misses. Applications include: protein shape classification (topology of backbone loops), brain connectivity analysis (persistent cycles in fMRI graphs), and point cloud segmentation (connected components at multiple scales).

Differentiable persistent homology. Gabrielsson et al. (2020) and Hofer et al. (2019) propose differentiable persistence layers: given a parameterised point cloud (or graph edge weights), backpropagate through the filtration by computing gradients via the implicit function theorem. This enables learning tasks such as: *topology optimisation* (create point clouds with target Betti numbers) and *topological regularisation* (penalise spurious loops in learned representations).

34.4 Riemannian Gradient Descent

On a Riemannian manifold $(\mathcal{M}, g)$, the gradient of $f : \mathcal{M} \to \mathbb{R}$ at $\mathbf{x}$ is the tangent vector $\operatorname{grad}_{\mathcal{M}f(\mathbf{x}) \in T_{\mathbf{x}}\mathcal{M}}$ defined by $g_{\mathbf{x}}(\operatorname{grad}_{\mathcal{M}f,\mathbf{v})=\mathrm{d}f(\mathbf{x})[\mathbf{v}]}$ for all $\mathbf{v} \in T_{\mathbf{x}}\mathcal{M}$. In coordinates, $\operatorname{grad}_{\mathcal{M}f=\mathbf{G}^{-1}\nabla f}$ where $\mathbf{G}_{ij} = g(\partial_i, \partial_j)$.

Riemannian gradient descent. One step moves along the geodesic in the negative gradient direction:

$$\mathbf{x}_{t+1} = \operatorname{Exp}_{\mathbf{x}_t}(-\alpha \operatorname{grad}_{\mathcal{M}f(\mathbf{x}_t))}$$

where $\operatorname{Exp}_{\mathbf{x}} : T_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$ is the Riemannian exponential map. The retraction $R_{\mathbf{x}}$ (a first-order approximation to $\operatorname{Exp}_{\mathbf{x}}$) is often used in practice when the exact geodesic is expensive.

Poincaré Ball Model

The Poincaré ball $\mathbb{B}_c^n = \{\mathbf{x} \in \mathbb{R}^n : c\|\mathbf{x}\|^2 < 1\}$ with metric $g_{\mathbf{x}} = \lambda_{\mathbf{x}}^2 \mathbf{I}$, where $\lambda_{\mathbf{x}} = \frac{2}{1-c\|\mathbf{x}\|^2}$, is a model of hyperbolic space with curvature $-c$. Geodesic distance:

$$d_c(\mathbf{x}, \mathbf{y}) = \frac{2}{\sqrt{c}} \tanh^{-1}(\sqrt{c}\,\|-\mathbf{x} \oplus_c \mathbf{y}\|)$$

where $\oplus_c$ is the Möbius addition. Points near the boundary $\|\mathbf{x}\| \to 1/\sqrt{c}$ are exponentially far from the origin — ideal for embedding deep trees with low distortion.

Riemannian Adam. Becigneul & Ganea (2019) extend Adam to manifolds: compute the Euclidean gradient ∇f, project to the tangent space via $\operatorname{grad}_{\mathcal{M}f=\lambda_{\mathbf{x}}^{-2}\nabla f}$ (for the Poincaré ball), accumulate first and second moment estimates in the tangent space, and retract back to the manifold after each update. In practice, this requires only replacing the Euclidean update step with a Möbius addition.

Convergence. For geodesically convex f on a Hadamard manifold (simply connected, non-positive curvature), Riemannian GD with step $\alpha = O(1/\sqrt{T})$ satisfies:

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\text{grad}_{\mathcal{M}} f(\mathbf{x}_t)\|^2 \leq \frac{2(f(\mathbf{x}_0) - f^*)}{\alpha T} + \frac{\alpha L^2}{2}$$

The same $O(1/\sqrt{T})$ rate as Euclidean GD, under Lipschitz gradient constant L on $\mathcal{M}$.

Applications in deep learning.

- *Hyperbolic embeddings*: Nickel & Kiela (2017) embed WordNet nouns in $\mathbb{B}^2$ with distortion $< 5\%$, vs. $> 35\%$ for Euclidean embeddings of the same dimension.
- *Hyperbolic GNNs*: replace flat node features with Poincaré ball coordinates; use Riemannian Adam for training.
- *SPD matrix networks*: networks operating on symmetric positive-definite (SPD) matrices (covariance descriptors in video understanding) use the affine-invariant metric and Riemannian GD on the SPD manifold.

Example — Riemannian Adam on Poincaré Embeddings

Training a 2D Poincaré embedding of WordNet hypernyms (82K nouns, 743K pairs): initialise all embeddings near the origin ($\|\mathbf{x}\| < 0.001$). Use Riemannian Adam with $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. After 200K steps (batch size 10, 10 negatives), embeddings organise hierarchically: abstract concepts (entity, object) cluster near the origin; specific terms (border collie, dalmatian) sit near the boundary. Reconstruction AP@10: 0.876 in $\mathbb{B}^2$ vs. 0.573 for 200-dim Euclidean word2vec. The Poincaré model uses $200\times$ fewer parameters.

Riemannian batch normalisation on the SPD manifold Batch normalisation on Euclidean layers computes mean and variance per channel. For SPD matrix networks (covariance descriptors), *Riemannian batch normalisation* (Brooks et al., 2019) computes the Fréchet mean $\bar{P} = \arg\min \sum_k d_g(P_k, P)^2$ (Riemannian mean) and normalises via parallel transport: $\hat{P}_k = \text{Exp}_{\bar{P}}(\alpha \,\text{Log}_{\bar{P}}(P_k) + \beta)$ where α, β are learnable scalars. This preserves the SPD structure (no invalid covariances) and improves convergence on EEG brain-computer interface classification ($+3\%$ accuracy over Euclidean BN applied to the vectorised SPD matrix).

34.5 Topological Data Analysis and Persistence

Topological Data Analysis (TDA) studies the shape of data through algebraic-topological invariants, providing descriptors that are complementary to geometric and statistical features. The central tool is persistent homology (§34.3); here we discuss how TDA integrates into machine learning pipelines.

Betti numbers. The k-th Betti number β_k counts k-dimensional "holes": β_0 = connected components, β_1 = independent loops, β_2 = enclosed voids. For a torus: $\beta_0 = 1$, $\beta_1 = 2$, $\beta_2 = 1$. Betti numbers are topological invariants: homeomorphic spaces have the same Betti numbers, but the converse is not true.

Euler characteristic. The alternating sum $\chi = \sum_k (-1)^k \beta_k$ equals the combinatorial Euler characteristic $V - E + F$ (vertices minus edges plus faces) for a simplicial complex — a

useful global invariant computable from the persistence diagram without full homology computation.

Mapper algorithm. Carlsson & Mémoli (2009) propose the Mapper algorithm: (1) apply a filter function $f : \mathcal{X} \to \mathbb{R}$ (e.g. PCA projection); (2) cover the range with overlapping intervals; (3) cluster points in each pre-image; (4) connect clusters sharing points. The resulting graph summarises the topological structure of the dataset. Mapper has been used to discover a subgroup of breast cancer patients with 100% survival rate (Nicolau et al., 2011) that was invisible to standard clustering.

Topological Regularisation in Deep Learning

Chen et al. (2019) add a persistence-based regulariser to the training loss:

$$\mathcal{L}_{\text{topo}} = \sum_{(b,d) \in \text{dgm}(\hat{f}) \setminus \text{dgm}(f^*)} (d - b)^2$$

This penalises topological features (persistence pairs) of the learned function $\hat{f}$ that are absent in the target topology $\text{dgm}(f^*)$. Gradients flow through the birth/death times via the implicit function theorem on the filtration. Applications: enforcing that a segmentation mask has exactly k connected components; training autoencoders whose latent space has the correct topology.

TDA on graphs. For graph-level classification, persistence diagrams are computed on the filtration defined by node/edge features (e.g. sorted by degree). Hofer et al. (2020) combine GNN-learned node features with differentiable persistence diagrams, backpropagating through both simultaneously. This GNN+TDA hybrid achieves state-of-the-art on graph classification benchmarks where purely spectral or purely structural methods fail.

Example — Persistent homology on protein contact graphs

Build a filtration on a protein contact map: $C_{ij} = 1$ if $\|\mathbf{r}_i - \mathbf{r}_j\| < \epsilon$ (Angstroms). Increase ϵ from 0 to 12Å; at each threshold add the corresponding contacts as edges. The H_0 persistence diagram records the merging of connected components (protein domains); H_1 captures loops (helix-helix contacts forming a ring). For CATH domain classification (1375 proteins, 4 classes), appending 20-dimensional persistence image features to GNN node features improves accuracy by $+2.1\%$ over GNN alone, demonstrating complementarity between local (GNN) and global (topological) structural descriptors.

35.1 The Five G's of Geometric Deep Learning

Bronstein et al. (2021) propose a unifying framework — *Geometric Deep Learning* (*GDL*) — deriving most deep learning architectures from group symmetry principles. The core idea: if the data domain Ω has a symmetry group $\mathfrak{G}$ acting on it, the model should be equivariant to that action. Five geometric domains cover the main cases in practice.

The Five G's.

1. **Grids** ($\Omega = \mathbb{Z}^d$, $\mathfrak{G} =$ translation): CNNs implement discrete translation-equivariant convolution. Stationarity of natural images is the inductive bias.

2. **Groups** ($\Omega = \mathfrak{G}$): the domain itself is a group (e.g. $SO(3)$, $SE(3)$). Group convolution

$(f \star h)(g) = \int_{\mathfrak{G}} f(u)h(u^{-1}g)\,du$ is equivariant to left-multiplication by $\mathfrak{G}$.

3. **Geodesics** ($\Omega = \mathcal{M},\ \mathfrak{G} =$ local isometries): intrinsic mesh/surface networks (MoNet, DiffusionNet). Convolution is defined via local geodesic patches.

4. **Graphs** ($\Omega = G,\ \mathfrak{G} = S_n$ permutations): GNNs implement permutation-equivariant message passing. The symmetry is the absence of canonical node ordering.

5. **Gauges** ($\Omega = \mathcal{M}$ with gauge symmetry): gauge-equivariant CNNs on manifolds (Cohen et al., 2019), relevant for physics-informed networks and climate modelling.

Blueprint: Equivariant Signal Processing

Given a domain Ω with symmetry group $\mathfrak{G}$ acting on signals $f : \Omega \to \mathbb{R}^d$, an *equivariant layer* Φ satisfies:

$$\Phi(L_g f) = L_g\,\Phi(f) \quad \forall\, g \in \mathfrak{G}$$

where $(L_g f)(x) = f(g^{-1} \cdot x)$ is the group action. *Invariant layers* satisfy $\Phi(L_g f) = \Phi(f)$ (e.g. global pooling). The GDL blueprint: identify $\mathfrak{G}$, construct the representation theory (irreps, Fourier transform), derive equivariant convolution via the convolution theorem, then compose equivariant layers into a deep network.

Deep learning architectures as special cases.

- CNNs: translation group on grids; spatial convolution is the equivariant operation.
- Spherical CNNs (Cohen et al., 2018): $SO(3)$ acting on S^2; spherical convolution via Wigner D-matrices.
- GNNs (all flavours): S_n acting on node sets; MPNN is the equivariant operation.
- Transformers: S_n acting on token sequences (permutation equivariant); attention is a special case of set functions.
- E(3)-equivariant networks (EGNN, NequIP): $E(3)$ acting on 3D point clouds.

The GDL framework predicts which symmetries remain to be exploited (e.g. time-reversal symmetry in physics, colour-permutation symmetry in image processing) and provides a principled route to designing new architectures for novel data domains.

Example — CNN as a special case of the GDL blueprint

Apply the GDL blueprint to 2D images: domain $\Omega = \mathbb{Z}^2$; symmetry group $\mathfrak{G} = \mathbb{Z}^2$ (translation); signal $f : \mathbb{Z}^2 \to \mathbb{R}^c$ (image). Equivariant operation: discrete convolution $(f \star \psi)(x) = \sum_y f(y)\psi(x - y)$, which satisfies $L_g(f \star \psi) = (L_g f) \star \psi$. Max-pooling is G-invariant (to the 2×2 translation sub-group). A CNN is therefore a depth composition of translation-equivariant layers (convolutions) followed by a translation-invariant readout (global pooling). Adding rotation symmetry ($\mathfrak{G} = p4$) yields a G-CNN; adding scale symmetry yields a scale-equivariant CNN. The blueprint predicts: if the data is also scale-invariant (e.g. histopathology), a scale-equivariant CNN should outperform a standard CNN without augmentation.

35.2 Message Passing Neural Networks (MPNN)

The MPNN framework (Gilmer et al., 2017) provides a unified view of most graph neural networks. A GNN is an MPNN if it can be written in the form:

MPNN: Message Passing and Readout

Message passing phase (for T steps). At step t, node i aggregates messages from neighbours:

$$\mathbf{m}_i^{(t)} = \sum_{j \in \mathcal{N}(i)} M_t\Big(\mathbf{h}_i^{(t)}, \mathbf{h}_j^{(t)}, e_{ij}\Big)$$

$$\mathbf{h}_i^{(t+1)} = U_t\Big(\mathbf{h}_i^{(t)}, \mathbf{m}_i^{(t)}\Big)$$

Readout phase. After T steps, compute a graph-level representation:

$$\hat{y} = R\Big(\Big\{\mathbf{h}_i^{(T)} : i \in \mathcal{V}\Big\}\Big)$$

M_t is the *message function*, U_t the *update function*, R the *readout function*.

Instantiations of the MPNN framework.

- *GCN* (Kipf & Welling, 2017): $M_t(\mathbf{h}_i, \mathbf{h}_j, e_{ij}) = \frac{1}{\sqrt{d_i d_j}}\mathbf{W}\mathbf{h}_j$; $U_t(\mathbf{h}, \mathbf{m}) = \sigma(\mathbf{m})$.
- *GAT* (Veličković et al., 2018): $M_t = \alpha_{ij}\mathbf{W}\mathbf{h}_j$ with attention-weighted messages; $U_t = \sigma(\mathbf{m})$.
- *GraphSAGE*: $M_t = \mathbf{W}_2\mathbf{h}_j$; $U_t = \sigma(\mathbf{W}_1[\mathbf{h}_i \,\|\, \text{MEAN}_j\{M_t\}])$.
- *GGNN* (Gated GNN): U_t is a GRU cell; enables variable-depth message passing with shared weights across steps.
- *MPNN for molecules* (original paper): edge-conditioned message $M_t(\mathbf{h}_i, \mathbf{h}_j, e_{ij}) = \mathbf{A}(e_{ij})\mathbf{h}_j$ where $\mathbf{A}(e)$ is a learned edge-type matrix; $R = \sum_i \sigma(\mathbf{i}_1\mathbf{h}_i^{(T)}) \odot (\mathbf{i}_2\mathbf{h}_i^{(T)})$ (set2set readout).

Expressiveness. The Weisfeiler-Lehman (WL) test (§35.3) upper-bounds MPNN expressiveness: no MPNN with sum aggregation can distinguish graphs that the 1-WL test deems isomorphic. GIN achieves this upper bound by using $M_t = \mathbf{h}_j$ and $U_t = \text{MLP}((1+\varepsilon)\mathbf{h}_i + \sum_j \mathbf{h}_j)$.

Beyond sum aggregation. Mean aggregation conflates counts with distributions; max aggregation loses multiplicity. Janossy pooling uses a permutation-invariant MLP over multisets; Deep Sets shows that any permutation-invariant function on a multiset decomposes as $\rho(\sum_i \phi(\mathbf{h}_i))$ — the canonical form for sum-decomposable aggregation. Principal Neighbourhood Aggregation (PNA) combines multiple aggregators (sum, mean, max, std) with degree-scalers, empirically outperforming single-aggregator MPNNs.

Example — Molecular property prediction with MPNN

For the QM9 dataset (134K small molecules, 12 quantum properties), the original MPNN (Gilmer et al., 2017) uses: bond-type edge features $e_{ij} \in \mathbb{R}^{14}$; set2set readout with $T = 6$ message-passing steps; edge-network message function $M(h_i, h_j, e_{ij}) = h_j\mathbf{A}(e_{ij})$ where $\mathbf{A} : \mathbb{R}^{14} \to \mathbb{R}^{h \times h}$ is a small network. Achieves mean absolute error of

0.05 kcal/mol on atomisation energy, competitive with DFT.

Over-smoothing in deep MPNNs Stacking many MPNN layers causes all node representations to converge to the same vector — *over-smoothing*. Formally, for GCN: $\mathbf{h}^{(k)} = \hat{\mathbf{A}}^k \mathbf{h}^{(0)} \mathbf{W}^k$. The spectral radius of $\hat{\mathbf{A}}$ is 1 (for connected bipartite graphs) or < 1, so $\hat{\mathbf{A}}^k \to \mathbf{1}\square^\top$ (the stationary distribution of the random walk). After $k \approx \log n / \log(1/\lambda_2)$ layers, all representations become indistinguishable. Mitigations: (1) residual connections $\mathbf{h}^{(k)} = \mathbf{h}^{(k-1)} + \text{MPNN}(\mathbf{h}^{(k-1)})$; (2) jumping knowledge (JK-Net): concatenate representations from all layers; (3) DropEdge: randomly remove edges during training, slowing convergence; (4) limit depth to $K \leq 4$–6.

35.3 Expressive Power: The Weisfeiler-Lehman Test

The **Weisfeiler-Lehman (WL) graph isomorphism test** (also called colour refinement or 1-WL) provides a fundamental upper bound on the expressiveness of any MPNN with injective aggregation.

1-WL algorithm. Initialise each node v with a colour $c^{(0)}(v) = \text{hash}(\ell(v))$ (from node label). At each iteration k:

$$c^{(k)}(v) = \text{hash}\big(c^{(k-1)}(v),\ \{\{c^{(k-1)}(u) : u \in \mathcal{N}(v)\}\}\big)$$

where $\{\{\cdot\}\}$ denotes a multiset. Iterate until the colour partition stabilises. Two graphs are declared *non-isomorphic* if their colour histograms differ; *possibly isomorphic* otherwise. The test is not complete (there exist non-isomorphic graphs it cannot distinguish).

MPNN $\leq$ 1-WL. Xu et al. (2019) proved: any MPNN with aggregation AGG and update U cannot distinguish more pairs of graphs than 1-WL. Moreover, if the aggregation is injective over multisets, the MPNN is exactly as powerful as 1-WL. GIN achieves this: $U(\mathbf{h}_v, \text{AGG}) = \text{MLP}((1+\varepsilon)\mathbf{h}_v + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u)$ is injective (by the universal approximation theorem on multisets).

Graphs WL fails to distinguish. The classic example is two regular graphs of the same degree and size: e.g. the Rook's graph $K_3 \square K_3$ and the Paley graph of order 9 — both 4-regular on 9 nodes, indistinguishable by 1-WL (and hence by any GIN).

Beyond 1-WL: higher-order tests.

- *k-WL*: operates on k-tuples of nodes; strictly more expressive as k increases. 3-WL distinguishes all graphs distinguishable by any GNN operating on local sub-graph patterns.
- *OSAN, PPGN*: compute sums over induced sub-graphs of size k; polynomial-time, more expressive than 1-WL.
- *Random features*: adding independent random node features (before message passing) makes any MPNN almost-surely as powerful as 3-WL (Sato et al., 2021).

Example — Cycle counting as a WL-separation task

Determining whether a graph contains a 4-cycle or 6-cycle is beyond 1-WL (Arvind et al., 2020). Consequently, no GIN can count cycles of length ≥ 4. OSAN and δ-k-LWL can count cycles by operating on 4-tuples or by considering short random walks as

sub-graph features.

Turing completeness of MPNNs with sufficient depth Loukas (2020) showed that MPNNs with $O(n)$ layers, hidden dimension $O(n)$, and $O(\log n)$ bits of precision can simulate any distributed algorithm (in the LOCAL model of distributed computing), making them Turing complete in this sense. However, practical GNNs use ≤ 10 layers and fixed-width representations — far from this asymptotic regime. For tasks requiring global coordination (e.g. detecting bipartiteness, counting spanning trees), depth $\Omega(n)$ is necessary, placing them outside the reach of any bounded-depth GNN regardless of aggregation function.

35.4 Graph Transformers

Pure MPNNs are limited by the WL-test expressiveness ceiling and suffer from over-squashing (exponentially many nodes compressed into fixed-size representations at graph readout). Graph Transformers address this by combining the local inductive bias of message passing with the global context of full attention.

GPS: General, Powerful, Scalable Graph Transformer (Rampášek et al., 2022). Each GPS layer applies MPNN and Transformer in parallel, then combines:

$$\mathbf{h}_i^{\text{MPNN}} = \text{MPNN}\big(\mathbf{h}_i, \{\mathbf{h}_j : j \in \mathcal{N}(i)\}, \{e_{ij}\}\big)$$

$$\mathbf{h}_i^{\text{Attn}} = \text{MultiHeadAttn}\big(\mathbf{h}_i, \{\mathbf{h}_j : j \in \mathcal{V}\}\big)$$

$$\mathbf{h}_i^{\text{out}} = \text{FFN}\big(\mathbf{h}_i^{\text{MPNN}} + \mathbf{h}_i^{\text{Attn}}\big)$$

GPS uses linear attention (Performer or BigBird) to reduce the $O(n^2)$ full-attention cost to $O(n)$, enabling scaling to graphs with thousands of nodes.

Positional and structural encodings (PE/SE). Unlike sequences, graphs have no canonical node ordering. Graph Transformers rely on expressive node features to supply positional context:

- *Laplacian PE*: eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ of the normalised graph Laplacian, giving a k-dimensional embedding invariant to graph isomorphism (up to eigenvector sign).
- *Random Walk SE*: $\text{RW}_{i,t} = [\text{RW}^t]_{ii}$ (the probability of a t-step random walk returning to node i) captures local structure at multiple scales.
- *RWSE + SignNet*: Kreuzer et al. (2021) learn sign-invariant Laplacian PE via a small equivariant network, resolving the eigenvector sign ambiguity.

Expander Graph Propagation (EGP)

Over-squashing in MPNNs occurs because local topology limits information flow between distant nodes. EGP adds *virtual expander edges* — a sparse, low-diameter augmentation (e.g. using Ramanujan expanders) that provides short paths between all node pairs without destroying the local structure. Combined with an MPNN, EGP achieves $O(\log n)$-diameter message passing.

Tokenised graph Transformers. GraphGPS-VQ (Ma et al., 2023) treats each node as a token and applies a full causal Transformer. Sub-graph sampling provides batching; relative positional biases derived from shortest-path distances replace absolute positional encodings.

Achieves state-of-the-art on OGB-LSC (3.8M nodes, 33M edges) with a 100M-parameter model.

Comparison to pure MPNNs.

Property	MPNN	Graph Transformer
Complexity per layer	$O(\|\mathcal{E}\|)$	$O(n^2)$ or $O(n)$ (linear)
Long-range dependencies	Limited (k hops)	Global
Expressiveness	$\leq$ 1-WL	$>$ 1-WL (with PE)
Over-squashing	Yes	Mitigated
Training stability	High	Requires careful PE design

35.5 Summary of Volume III

This volume has covered the mathematics of structure, symmetry, and multi-modal alignment:

- **Part 25:** Graph Laplacian and spectrum, Dirichlet energy, Graph Fourier Transform, spectral gap, Cheeger's inequality, random walk and WL kernels.
- **Part 26:** Spectral convolution, ChebNet polynomial approximation, GCN derivation and spatial interpretation, GIN and expressiveness.
- **Part 27:** GAT attention coefficients, GATv2, multi-head GAT, GraphSAGE inductive aggregation, heterogeneous graphs and R-GCN.
- **Part 28:** Groups and symmetries, invariance vs. equivariance, group convolution, Schur's Lemma, Wigner D-matrices and spherical harmonics.
- **Part 29:** EGNN — $SE(3)$-equivariant message passing, coordinate update derivation, NequIP tensor features.
- **Part 30:** Optimal transport, Monge-Kantorovich, Kantorovich duality, Sinkhorn algorithm, Wasserstein barycenters.
- **Part 31:** CLIP and InfoNCE derivation, alignment-uniformity decomposition, cross-attention for vision-language fusion, Flamingo few-shot learning.
- **Part 32:** NeRF volume rendering equation, transmittance, sinusoidal positional encoding, spectral bias of MLPs, SIREN for PDEs.
- **Part 33:** AlphaFold MSA attention, Evoformer triangle updates, Invariant Point Attention, backbone frame update via $SE(3)$ composition, FAPE loss.
- **Part 34:** Smooth manifolds, geodesics, parallel transport, persistent homology, Riemannian gradient descent, hyperbolic GNNs.
- **Part 35:** Geometric deep learning blueprint, MPNN framework and comparison table, WL expressiveness, graph positional encodings, GPS hybrid architecture.

Volume IV will cover advanced optimisation theory, generalisation bounds, adversarial robustness, and differential privacy.

Looking ahead to Volumes IV and V The geometric deep learning frameworks developed here connect to the remaining volumes in two directions. *Volume IV* (Optimisation and Robustness) revisits GNN training dynamics: adversarial robustness of graph models (graph injection attacks, feature perturbation), differential privacy for node embeddings, and PAC-Bayes generalisation bounds that incorporate graph topology. *Volume V* (Efficient and Emergent Intelligence) extends the attention story: the state-space model (S4/Mamba) can be viewed as a graph signal processing operation on a chain graph with selective filters; mechanistic interpretability of GNNs and Transformers uses the same spectral and topological tools developed here.

36.1 Transformer Architecture: Full Mathematical Derivation

The transformer (Vaswani et al., 2017) is the dominant architecture for sequence modelling, replacing recurrence with multi-head self-attention. We derive the full mathematical formulation.

Input representation. A sequence of n tokens is embedded into $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{model}}}$. Positional encodings $\mathbf{P} \in \mathbb{R}^{n \times d_{\text{model}}}$ are added:

$$\mathbf{Z}^{(0)} = \mathbf{X} + \mathbf{P}, \qquad P_{t,2k} = \sin\left(\frac{t}{10000^{2k/d}}\right), \; P_{t,2k+1} = \cos\left(\frac{t}{10000^{2k/d}}\right)$$

The sinusoidal encoding assigns each position a unique frequency pattern. For rotary positional embeddings (RoPE), positions are encoded by rotating key-query products in 2D subspaces: $(\mathbf{q}_m, \mathbf{k}_n) \mapsto (\mathbf{q}_m e^{im\theta}, \mathbf{k}_n e^{in\theta})$, preserving relative position $m - n$ in the dot product.

Scaled dot-product attention. For queries $\mathbf{Q} \in \mathbb{R}^{n \times d_k}$, keys $\mathbf{K} \in \mathbb{R}^{n \times d_k}$, values $\mathbf{V} \in \mathbb{R}^{n \times d_v}$:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

The scaling by $1/\sqrt{d_k}$ prevents the dot products from growing large in magnitude (and hence the softmax from saturating to one-hot) when d_k is large. Formally, if $\mathbf{q}, \mathbf{k} \sim \mathcal{N}(0, \mathbf{I}_{d_k})$ independently, then $\mathbf{q} \cdot \mathbf{k} \sim \mathcal{N}(0, d_k)$; dividing by $\sqrt{d_k}$ restores unit variance.

Multi-head attention. H parallel attention heads, each projecting to subspaces:

$$\mathbf{Q}_h = \mathbf{Z}\mathbf{W}_h^Q, \quad \mathbf{K}_h = \mathbf{Z}\mathbf{W}_h^K, \quad \mathbf{V}_h = \mathbf{Z}\mathbf{W}_h^V, \qquad \mathbf{W}_h^Q, \mathbf{W}_h^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, \; \mathbf{W}_h^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

$$\text{MultiHead}(\mathbf{Z}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\mathbf{W}^O, \qquad \text{head}_h = \text{Attention}(\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h)$$

With H heads and $d_k = d_v = d_{\text{model}}/H$, total parameters for one MHA layer: $4d_{\text{model}}^2$ (from $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V, \mathbf{W}^O$).

Layer norm and residual connections. Each transformer block applies:

$$\mathbf{Z}' = \mathbf{Z} + \text{MHA}(\text{LayerNorm}(\mathbf{Z}))$$

$$\mathbf{Z}'' = \mathbf{Z}' + \text{FFN}(\text{LayerNorm}(\mathbf{Z}'))$$

The feed-forward network is $\text{FFN}(\mathbf{x}) = \mathbf{W}_2\,\text{ReLU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2$ with inner dimension $d_{\text{ff}} = 4d_{\text{model}}$. The FFN accounts for $8d_{\text{model}}^2$ parameters — twice the MHA — and is responsible for "memorising" factual associations.

Complexity analysis. Attention cost: $O(n^2 d_k)$ time and $O(n^2)$ memory (the attention matrix). For $n = 4096$, $d_k = 64$: the attention matrix alone is $4096^2 \times 4 \approx 64$ MB per head. Flash Attention (Dao et al., 2022) reduces the memory footprint to $O(n)$ via tiled computation, achieving 2–4× speedup by maximising hardware SRAM utilisation.

Decoder and encoder-decoder variants. Encoder-only (BERT): bidirectional attention, masked language modelling pre-training. Decoder-only (GPT): causal (masked) attention, autoregressive generation. Encoder-decoder (T5, BART): encoder processes the source, decoder cross-attends to encoder output.

Example — Attention as information retrieval

Attention is a *soft dictionary lookup*: the query $\mathbf{q}$ retrieves a weighted average of values $\{\mathbf{v}_i\}$, with weights proportional to the similarity $\mathbf{q} \cdot \mathbf{k}_i / \sqrt{d_k}$. A hard dictionary retrieves the single most similar key; soft attention retrieves a smooth interpolation. The softmax temperature τ (implicit $1/\sqrt{d_k}$) controls sharpness: $\tau \to 0$ gives hard retrieval, $\tau \to \infty$ gives uniform averaging.

36.2 Rotary Positional Embedding (RoPE) and ALiBi

Absolute positional encodings (sinusoidal or learned) add a position vector $\mathbf{p}_t$ to the token embedding at position t. Both RoPE and ALiBi instead encode position *relative* between query and key, enabling better length generalisation.

RoPE: Rotary Position Embedding (Su et al., 2021). Encodes position m by rotating the query/key vectors in 2D sub-spaces. For a d-dimensional query split into $d/2$ pairs (q_{2i-1}, q_{2i}), define the rotation matrix:

$$\mathbf{R}_m^{(i)} = \begin{pmatrix} \cos m\theta_i & -\sin m\theta_i \\ \sin m\theta_i & \cos m\theta_i \end{pmatrix}, \quad \theta_i = 10000^{-2(i-1)/d}$$

Apply $\mathbf{R}_m^{(i)}$ to the i-th pair of $\mathbf{q}_m$ and $\mathbf{R}_n^{(i)}$ to the i-th pair of $\mathbf{k}_n$. The inner product:

$$\langle \mathbf{R}_m \mathbf{q}, \mathbf{R}_n \mathbf{k} \rangle = \mathbf{q}^\top \mathbf{R}_{n-m} \mathbf{k}$$

depends only on the *relative* position $n - m$, not on m or n individually — rotation is a symmetry of the inner product. Different frequency bands θ_i encode position at multiple scales, analogous to the sinusoidal encoding but in a multiplicative (rotation) rather than additive form.

Long-context extension. RoPE generalises to positions beyond the training length by adjusting the base frequency: *YaRN* (Peng et al., 2023) scales θ_i non-uniformly — high-frequency dimensions (small i) are barely modified, while low-frequency dimensions (large i, which encode coarse position) are scaled down, allowing extrapolation to $4\times$ training length with minimal perplexity degradation.

ALiBi: Attention with Linear Biases

Press et al. (2022) add a fixed, non-learned linear penalty to attention logits:

$$a_{ij} = \mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d_k} - m_h \cdot |i - j|$$

where $m_h > 0$ is a head-specific slope (geometric sequence: $m_h = 2^{-8h/H}$ for H heads). No positional vectors are added; the bias directly penalises attending to distant tokens. ALiBi extrapolates to longer sequences than trained on with less perplexity increase than RoPE (tested to $5\times$ training length). Trade-off: slightly worse performance at training-length sequences.

Comparison.

Method	Relative?	Length generalisation
Sinusoidal	No	Poor
Learned abs. PE	No	Poor
RoPE	Yes	Good (2–4× with YaRN)
ALiBi	Yes (bias)	Very good (5×+)

RoPE is the dominant choice in current LLMs (LLaMA 2/3, Mistral, Qwen, Gemma); ALiBi is used in MPT and BLOOM. Flash Attention 2 has native support for both.

Example — RoPE interpolation for 32K-context LLaMA

LLaMA 2 was trained with maximum context 4096. To extend to 32768 tokens: apply *position interpolation* (Chen et al., 2023) — rescale all positions by $4096/32768 = 1/8$

before computing RoPE, then fine-tune on 1000 steps of long-context data ($\sim 0.1\%$ of pre-training compute). Perplexity at 32K is < 15 (compared to $> 10^6$ without interpolation and > 80 with naive extrapolation). The intuition: interpolation asks the model to distinguish positions m and $m+1$ that are now $1/8$ as far apart in frequency space — more difficult but manageable; extrapolation asks it to handle phases it has never seen.

36.3 Flash Attention: Hardware-Aware Algorithm

Standard attention materialises the full $n \times n$ attention matrix $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top/\sqrt{d_k}$ in GPU HBM (high-bandwidth memory), then applies softmax and multiplies by $\mathbf{V}$. For $n = 8192$, $H = 32$ heads: $\mathbf{S}$ alone requires $32 \times 8192^2 \times 4\,\text{bytes} \approx 8\,\text{GB}$ — exceeding the SRAM of any GPU and forcing repeated HBM reads/writes.

Tiled computation. Flash Attention (Dao et al., 2022) tiles $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ into blocks that fit in SRAM ($\sim 20\,\text{MB}$ on A100), computes partial softmax results tile-by-tile, and accumulates the output without ever writing the full attention matrix to HBM:

$$\text{For blocks } \mathbf{K}_j, \mathbf{V}_j: \quad \tilde{\mathbf{S}}_{ij} = \mathbf{Q}_i \mathbf{K}_j^\top / \sqrt{d_k}$$

The online softmax trick (Milakov & Gimelshein, 2018) maintains running max $m^{(j)}$ and normaliser $\ell^{(j)}$ to update the output accumulator $\mathbf{O}_i$ incrementally:

$$m_i^{(j)} = \max(m_i^{(j-1)}, \text{rowmax}(\tilde{\mathbf{S}}_{ij})), \quad \ell_i^{(j)} = e^{m_i^{(j-1)} - m_i^{(j)}} \ell_i^{(j-1)} + \text{rowsum}(e^{\tilde{\mathbf{S}}_{ij} - m_i^{(j)}})$$

$$\mathbf{O}_i^{(j)} = \text{diag}(e^{m_i^{(j-1)} - m_i^{(j)}}) \mathbf{O}_i^{(j-1)} + e^{\tilde{\mathbf{S}}_{ij} - m_i^{(j)}} \mathbf{V}_j$$

After all j: $\mathbf{O}_i = \text{diag}(\ell_i^{(J)})^{-1} \mathbf{O}_i^{(J)}$.

Flash Attention Complexity

- **FLOPs**: $O(n^2 d)$ — identical to standard attention. No asymptotic reduction in arithmetic.
- **HBM reads/writes**: $O(n^2/M)$ passes over $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ of size $O(nd)$, vs. $O(n^2)$ for standard attention. With SRAM size M: total IO $= O(nd \cdot n/M) = O(n^2 d/M)$.
- **Additional memory**: $O(n)$ (stores only m_i, ℓ_i, $\mathbf{O}_i$ blocks).
- **Wall-clock speedup**: 2–4× on A100 for $n = 2048$; up to 7× for $n = 16384$.

Flash Attention 2. Dao (2023) further reduces non-matmul FLOPs (the online softmax rescaling) and reorganises work across thread blocks to achieve $\approx 70\%$ of A100 peak TFLOP/s, vs. $\approx 30\%$ for Flash Attention 1.

Flash Attention 3. Targets H100 Hopper architecture: uses hardware-supported FP8 precision and overlaps GEMM/softmax stages via asynchronous WGMMA instructions, reaching ≈ 1.2 PFLOP/s on FP16.

Backward pass. Rather than storing the $O(n^2)$ attention matrix for the backward pass, Flash Attention recomputes it on-the-fly during backpropagation from the stored (m_i, ℓ_i) statistics — a form of gradient checkpointing that trades HBM memory for extra FLOPs.

Flash Attention and the roofline model GPU performance is modelled by the roofline: an operation is either *compute-bound* (peak TFLOP/s limits speed) or *memory-*

bandwidth-bound (peak HBM bandwidth limits speed). For $n = 2048$, $d_k = 64$: attention FLOPs $= 4n^2 d_k = 10^9$; HBM bytes for $\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{S}, \mathbf{P}, \mathbf{O} = 6n^2 \cdot 4 = 100$MB. On A100 (312 TFLOP/s, 2TB/s bandwidth): compute time $= 3.2\mu$s; memory time $= 50\mu$s. Standard attention is $15\times$ memory-bandwidth-bound. Flash Attention eliminates the $\mathbf{S}$ and $\mathbf{P}$ HBM writes (saving $2n^2 \cdot 4 = 50$MB) and halves HBM traffic, making attention nearly compute-bound — explaining the 2–4× empirical speedup.

36.4 Sparse and Linear Attention Variants

Standard self-attention costs $O(n^2 d)$ time and $O(n^2)$ memory. For long sequences ($n > 4096$), this is prohibitive. Several lines of work reduce this to $O(n)$ or $O(n \log n)$.

Sparse attention patterns. Restrict each token to attend to a fixed subset $S_i \subset [n]$, $|S_i| \ll n$:

- *Local window attention*: each token attends to a window of w neighbours. Cost $O(nwd)$. Used in Longformer.
- *Strided attention*: tokens at stride s attend globally. Combined with local attention in BigBird: $O(n)$ with theoretical expressiveness matching full attention for certain tasks.
- *Axial attention*: for 2D grids, attend along rows then columns separately. Cost $O(n\sqrt{n}\, d)$.

Linear attention. Replace softmax attention with a kernel decomposition. Standard attention:

$$\text{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})_i = \frac{\sum_j \exp(\mathbf{q}_i \cdot \mathbf{k}_j)\mathbf{v}_j}{\sum_j \exp(\mathbf{q}_i \cdot \mathbf{k}_j)}$$

Approximate $\exp(\mathbf{q} \cdot \mathbf{k}) \approx \phi(\mathbf{q})^\top \phi(\mathbf{k})$ for a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^r$. Then:

$$\text{LinAttn}_i = \frac{\phi(\mathbf{q}_i)^\top \left(\sum_j \phi(\mathbf{k}_j)\mathbf{v}_j^\top\right)\mathbf{1}}{\phi(\mathbf{q}_i)^\top \left(\sum_j \phi(\mathbf{k}_j)\right)}$$

The matrix $\sum_j \phi(\mathbf{k}_j)\mathbf{v}_j^\top \in \mathbb{R}^{r \times d}$ is computed once; each query applies it in $O(rd)$. Total cost: $O(nrd)$.

Performer (FAVOR+)

Choromanski et al. (2021) construct ϕ using random Fourier features that *unbiasedly approximate* the softmax kernel: $\mathbb{E}[\phi(\mathbf{q})^\top \phi(\mathbf{k})] = \exp(\mathbf{q} \cdot \mathbf{k}/\sqrt{d})$.

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{r}} \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sqrt{d}}\right) [\exp(\square_1 \cdot \mathbf{x}), \ldots, \exp(\square_r \cdot \mathbf{x})]$$

with $\square_i \sim \mathcal{N}(0, \mathbf{I}_d)$ (or orthogonal random features for lower variance). Performer achieves $O(nrd)$ attention with provable approximation guarantees and strong empirical performance on long-context tasks.

Flash Attention recap. Flash Attention (Dao et al., 2022) does *not* reduce the asymptotic FLOPs of exact attention ($O(n^2 d)$) but reorganises the computation into tiles that fit in the GPU's SRAM (fast on-chip memory), avoiding $O(n^2)$ reads/writes to HBM. This gives 2–4× wall-clock speedup and $O(n)$ additional memory (no need to materialise the full attention matrix). Flash Attention 2 further optimises by minimising non-matrix-multiplication FLOPs, achieving $\approx 70\%$ of theoretical peak A100 throughput.

Sliding-window attention (Mistral). Mistral 7B uses a sliding window of size $w = 4096$ with rotary positional embeddings and grouped-query attention (GQA). GQA shares key-value heads across query heads, reducing the KV-cache memory from $O(n \cdot H)$ to $O(n \cdot G)$ where $G \ll H$ is the number of KV groups. This enables longer inference sequences without increasing memory costs.

Example — Longformer on long document classification

Beltagy et al. (2020) apply Longformer (sliding window + global attention) to the HotpotQA multi-hop reasoning task (average 600 tokens) and TRIVIAQA (average 2000 tokens). Local window $w = 512$; global attention on question tokens. Longformer achieves $+2.3$ F1 over RoBERTa (truncated to 512 tokens) on HotpotQA, as the longer context provides evidence from multiple paragraphs. Memory: $O(nw)$ vs. $O(n^2)$ for full attention — enables $n = 4096$ on a single 16GB GPU where full attention would require 64GB.

37.1 Score-Based Diffusion Models: Mathematical Framework

Score-based diffusion models define a continuous-time process that gradually adds noise to data and learns to reverse it.

Forward SDE. The data $\mathbf{x}_0 \sim p_{\text{data}}$ is gradually noised via an Itô stochastic differential equation:

$$d\mathbf{x} = f(\mathbf{x}, t)\, dt + g(t)\, d\mathbf{w}$$

where $\mathbf{w}$ is a standard Wiener process (Brownian motion), $f(\cdot, t) : \mathbb{R}^d \to \mathbb{R}^d$ is the drift coefficient, and $g(t) : \mathbb{R} \to \mathbb{R}$ is the diffusion coefficient. For the variance-preserving (VP-SDE) process: $f(\mathbf{x}, t) = -\frac{1}{2}\beta(t)\mathbf{x}$, $g(t) = \sqrt{\beta(t)}$, with $\beta(t)$ increasing from β_{min} to β_{max}.

Marginal distribution. Under VP-SDE, the conditional marginal $p_{0t}(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \alpha(t)\mathbf{x}_0, \sigma^2(t)\mathbf{I})$ where:

$$\alpha(t) = e^{-\frac{1}{2}\int_0^t \beta(s)\, ds}, \qquad \sigma^2(t) = 1 - e^{-\int_0^t \beta(s)\, ds}$$

As $t \to T$, $\alpha(t) \to 0$ and $\sigma^2(t) \to 1$, so $p_T \approx \mathcal{N}(0, \mathbf{I})$.

Score function. The *score* of the noised distribution is $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$. By Tweedie's formula, for the Gaussian marginal:

$$\nabla_{\mathbf{x}} \log p_t(\mathbf{x}_t|\mathbf{x}_0) = -\frac{\mathbf{x}_t - \alpha(t)\mathbf{x}_0}{\sigma^2(t)} = -\frac{\epsilon 0}{\sigma(t)}$$

where $\epsilon 0 \sim \mathcal{N}(0, \mathbf{I})$ is the added noise. A neural network $\epsilon 0_\theta(\mathbf{x}_t, t)$ is trained to predict this noise (equivalent to score matching).

Reverse SDE. Anderson (1982) showed the reverse of an Itô SDE is:

$$d\mathbf{x} = [f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\, dt + g(t)\, d\bar{\mathbf{w}}$$

where $\bar{\mathbf{w}}$ is a reverse-time Wiener process. Substituting the learned score $s_\theta(\mathbf{x}, t) \approx \nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ gives a trainable generative model.

Probability flow ODE. Every SDE has an equivalent probability-flow ODE with the same marginals:

$$d\mathbf{x} = \left[f(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt$$

This is a deterministic ODE that can be solved efficiently with adaptive-step ODE solvers (DOPRI5), enabling fast sampling with fewer function evaluations than DDPM.

Example — DDPM as a discretised VP-SDE

DDPM (Ho et al., 2020) discretises the VP-SDE with $T = 1000$ steps. The noise schedule $\beta_t \in [10^{-4}, 0.02]$ corresponds to $\alpha_t = \prod_{s=1}^{t}(1-\beta_s)$. Training objective:

$$\mathcal{L} = \mathbb{E}_{t,\mathbf{x}_0,\epsilon 0}\left[\|\epsilon 0 - \epsilon 0_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon 0, t)\|^2\right]$$

At test time, denoise from $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$ via $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon 0_\theta) + \sigma_t \mathbf{z}$.

Score matching and Fisher divergence The *score function* $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ is the direction of steepest ascent of the log-density. Hyvärinen (2005) showed that the Fisher divergence $\mathbb{E}_p[\|s_\theta(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{x})\|^2]$ can be minimised without knowing $p(\mathbf{x})$ explicitly, using *implicit score matching*: $\mathbb{E}_p[\mathrm{tr}(\nabla_{\mathbf{x}} s_\theta(\mathbf{x})) + \frac{1}{2}\|s_\theta(\mathbf{x})\|^2]$. The trace term requires computing the divergence of s_θ, costing $O(d)$ backward passes. *Sliced score matching* projects onto random directions, reducing cost to $O(1)$. Denoising score matching (Vincent, 2011) avoids both issues: matching scores at each noise level σ_t is equivalent to the denoising objective.

Example — Score estimation on a mixture of Gaussians

For $p(\mathbf{x}) = \frac{1}{2}\mathcal{N}(\mathbf{x}; \square_1, \mathbf{I}) + \frac{1}{2}\mathcal{N}(\mathbf{x}; \square_2, \mathbf{I})$ with $\|\square_1 - \square_2\| = 5$: the true score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ is a mixture of two linear functions, pointing towards $\square_1$ from the first mode and towards $\square_2$ from the second, with a complex transition near the boundary. Denoising score matching trains $s_\theta(\mathbf{x}, \sigma)$ at noise level $\sigma = 0.1$ and achieves MSE < 0.01 on a test grid, accurately capturing both modes' score fields. At $\sigma = 3$ (high noise), the score is nearly linear — easier to learn but loses multimodal structure. The optimal σ schedule (from $\sigma_T = 3$ down to $\sigma_1 = 0.01$) is designed so each step is tractable and the aggregate covers all scales.

37.2 Flow Matching and Rectified Flows

Flow matching (Lipman et al., 2022; Liu et al., 2022) provides a simulation-free training framework for *continuous normalising flows* (*CNFs*), learning an ODE vector field that maps a source distribution to a target distribution without the costly ODE simulation required by standard CNF training.

Conditional flow matching. Define a probability path $p_t(\mathbf{x})$ interpolating between $p_0 = \mathcal{N}(0, \mathbf{I})$ and $p_1 = p_{\text{data}}$ via:

$$\mathbf{x}_t = \psi_t(\mathbf{x}_0|\mathbf{x}_1) = (1 - (1 - \sigma_{\min})t)\mathbf{x}_0 + t\mathbf{x}_1$$

a linear interpolation (affine path). The marginal vector field generating p_t is:

$$u_t(\mathbf{x}) = \mathbb{E}\left[u_t(\mathbf{x}|\mathbf{x}_1) \,\middle|\, \mathbf{x}_t = \mathbf{x}\right], \quad u_t(\mathbf{x}|\mathbf{x}_1) = \frac{\mathbf{x}_1 - (1 - \sigma_{\min})\mathbf{x}_0}{1 - (1 - \sigma_{\min})t}$$

Training a network v_θ to match the *conditional* vector field is equivalent to matching the marginal field, with loss:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t, \mathbf{x}_0, \mathbf{x}_1}\left[\|v_\theta(\mathbf{x}_t, t) - u_t(\mathbf{x}_t|\mathbf{x}_1)\|^2\right]$$

No ODE simulation during training — only evaluating v_θ at a single point per sample.

Rectified Flow (ReFlow)

Liu et al. (2022) show that after one training pass, the ODE trajectories are generally curved. *Reflow*: generate paired samples $(\mathbf{x}_0, \mathbf{x}_1)$ by running the ODE forward with the trained v_θ, then retrain on the new pairs. Reflow straightens the trajectories (reduces curvature), enabling high-quality generation in fewer ODE steps — ultimately achieving 1-step generation with negligible quality loss after 2 reflow iterations.

Optimal transport (OT-CFM). Tong et al. (2023) pair $\mathbf{x}_0 \sim p_0$ and $\mathbf{x}_1 \sim p_1$ using the *mini-batch OT map* (§30.1–§30.3), minimising transport cost within each batch. This gives

straighter trajectories by construction (OT-paired samples require less "crossing" in transport), reducing the NFE (number of function evaluations) at inference from ~ 100 (flow matching) to ~ 10 (OT-CFM).

Connection to diffusion models. Setting $\sigma_{\min} \to 0$ and using the VP-SDE noise schedule recovers DDPM as a special case of flow matching. Flow matching is strictly more general: it supports any interpolant (affine, cosine, VP, VE) and any pairing strategy (independent, OT, data-conditioned). Stable Diffusion 3 (Esser et al., 2024) uses rectified flow matching, achieving state-of-the-art text-to-image generation with $2\times$ fewer sampling steps than DDIM.

> Connection to CNFs and continuous NFs A *continuous normalising flow* (CNF, Chen et al., 2018) defines a probability path via an ODE $d\mathbf{x}_t/dt = v_\theta(\mathbf{x}_t, t)$ with $\mathbf{x}_0 \sim p_0$ and $\mathbf{x}_1 \sim p_{\text{data}}$. Training a CNF requires simulating the ODE at each iteration (to compute the change-of-variables log-determinant), costing $O(T_{\text{steps}} \cdot d)$ per sample. Flow matching removes this bottleneck by providing a direct regression target $u_t(\mathbf{x}_t|\mathbf{x}_1)$ at each $(t, \mathbf{x}_t)$, enabling single-step gradient computation. The CNF vector field is fully determined by the regression solution; ODE simulation is needed only at inference.

37.3 Classifier-Free Guidance and DDIM Sampling

Classifier-free guidance (CFG). Standard diffusion conditioning requires a separate classifier $p_\phi(c|\mathbf{x}_t)$ to guide the reverse process (classifier guidance). CFG (Ho & Salimans, 2022) eliminates this by jointly training a single model for both conditional and unconditional generation:

During training, the conditioning $\mathbf{c}$ is randomly set to a null token $\emptyset$ with probability $p_{\text{uncond}} = 0.1$–0.2. At inference, the score is linearly extrapolated:

$$\tilde{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{c}) = \epsilon_\theta(\mathbf{x}_t, t, \emptyset) + s\big[\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c}) - \epsilon_\theta(\mathbf{x}_t, t, \emptyset)\big]$$

with guidance scale $s > 1$. Interpretation: $\tilde{\epsilon}$ moves in the direction of $\epsilon(\cdot, \mathbf{c}) - \epsilon(\cdot, \emptyset)$ — towards regions where the conditional score exceeds the unconditional score. Larger s sharpens adherence to $\mathbf{c}$ at the cost of sample diversity (a precision/recall trade-off).

Optimal guidance. The true classifier-guided score is $\nabla_{\mathbf{x}_t} \log p(\mathbf{c}|\mathbf{x}_t) = \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t|\mathbf{c}) - \nabla_{\mathbf{x}_t} \log p_\theta(\mathbf{x}_t)$. CFG approximates this without training a noisy classifier, and empirically achieves better FID-CLIP trade-offs than explicit classifier guidance.

DDIM: Deterministic Sampling (Song et al., 2021). The DDPM reverse process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\square_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$ requires $T = 1000$ steps for good quality. DDIM replaces the Markov assumption with a non-Markovian forward process whose marginals $q(\mathbf{x}_t|\mathbf{x}_0)$ are identical to DDPM:

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \underbrace{\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}}_{\text{predicted } \hat{\mathbf{x}}_0} + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\,\epsilon_\theta(\mathbf{x}_t, t) + \sigma_t \mathbf{z}$$

Setting $\sigma_t = 0$ for all t gives a *deterministic* ODE trajectory, enabling:

- **Accelerated sampling**: skip to any sub-sequence of timesteps $\{t_1, \ldots, t_S\} \subset [T]$ with $S \ll T$. Quality at $S = 50$ steps rivals DDPM at $S = 1000$.

- **Latent encoding**: given a real image $\mathbf{x}_0$, run the ODE *forward* to obtain $\mathbf{x}_T$ (*DDIM inversion*). Then edit and decode — the determinism enables exact reconstruction.
- **Interpolation**: traverse straight paths in latent space $\mathbf{x}_T$ between two images.

DPM-Solver

Chen et al. (2022) interpret the diffusion reverse ODE as a linear ODE with an exponential integrator and derive high-order (2nd/3rd order) multi-step solvers. DPM-Solver++ achieves near-exact DDPM quality at $S = 15$–20 function evaluations, enabling real-time (< 1s) image generation.

Guidance scale and precision/recall trade-off The FID vs. IS (Inception Score) trade-off in conditional generation is governed by the guidance scale s. At $s = 1$ (no guidance): high recall (diverse outputs), lower FID, lower IS. At $s = 7$: high precision (outputs match the prompt), higher IS, higher FID. Dhariwal & Nichol (2021) measure this as the "precision/recall" of the generated distribution: precision = fraction of generated samples falling in the manifold of real data; recall = fraction of the real data manifold covered by generated samples. Classifier-free guidance at $s > 1$ increases precision at the cost of recall — a fundamental trade-off, not a deficiency of any specific model.

37.4 Latent Diffusion and Stable Diffusion

Pixel-space diffusion models for high-resolution images are computationally expensive: forward and reverse passes both scale with the pixel count. Latent Diffusion Models (LDMs; Rombach et al., 2022) move the diffusion process to a compressed latent space, drastically reducing compute while preserving perceptual quality.

Architecture overview. LDM decouples generation into three components:

1. **Perceptual encoder/decoder** $(\mathcal{E}, \mathcal{D})$: a VQ-GAN or KL-regularised autoencoder compresses $\mathbf{x} \in \mathbb{R}^{H\times W\times 3}$ to $\mathbf{z} = \mathcal{E}(\mathbf{x}) \in \mathbb{R}^{h\times w\times c}$ with spatial downsampling factor $f = H/h \in \{4, 8, 16\}$.
2. **Latent diffusion model** $\epsilon_\theta(\mathbf{z}_t, t, \mathbf{c})$: a U-Net or Transformer operating on $\mathbf{z}_t$ at each diffusion step, conditioned on $\mathbf{c}$.
3. **Conditioning mechanism**: text, class labels, segmentation maps, or image embeddings are injected via cross-attention in each U-Net layer.

Training objective. Standard denoising score matching in latent space:

$$\mathcal{L}_{\text{LDM}} = \mathbb{E}_{\mathbf{z}\sim\mathcal{E}(\mathbf{x}),\,\epsilon\sim\mathcal{N}(0,\mathbf{I}),\,t}\left[\|\epsilon - \epsilon_\theta(\mathbf{z}_t, t, \mathbf{c})\|_2^2\right]$$

where $\mathbf{z}_t = \sqrt{\bar{\alpha}_t}\,\mathbf{z} + \sqrt{1-\bar{\alpha}_t}\,\epsilon$ is the noised latent. No reconstruction loss on pixels during diffusion training — the encoder/decoder handles pixel fidelity.

Stable Diffusion (SD)

Stable Diffusion is a specific LDM with: KL-regularised autoencoder ($f = 8$, latent $64\times64\times4$ for 512^2 images); U-Net with 880M parameters and cross-attention to CLIP text embeddings; DDIM or PLMS samplers for fast inference (20–50 steps). SD 1.x was trained on LAION-5B (5 billion image-text pairs). The latent U-Net reduces the

cost per step from $O(H^2W^2)$ (pixel attention) to $O((H/8)^2(W/8)^2)$ — a $64\times$ reduction in attention computation.

Classifier-free guidance at inference. Text-conditioned generation uses:

$$\tilde{\epsilon}_\theta(\mathbf{z}_t, t, \mathbf{c}) = \epsilon_\theta(\mathbf{z}_t, t, \emptyset) + s(\epsilon_\theta(\mathbf{z}_t, t, \mathbf{c}) - \epsilon_\theta(\mathbf{z}_t, t, \emptyset))$$

with guidance scale $s \in [7, 15]$ trading diversity for prompt adherence. A single model serves both roles by randomly dropping the condition $\mathbf{c}$ with probability $p = 0.1$ during training.

ControlNet and fine-grained conditioning. Zhang & Agrawala (2023) freeze the LDM weights and add trainable copies of the U-Net encoder blocks, connected to the frozen decoder via zero-initialised convolutional layers. This allows conditioning on structured inputs (depth maps, edge maps, human poses) without retraining the full model, enabling precise spatial control over generation.

DiT: Diffusion Transformers Peebles & Xie (2023) replace the U-Net backbone of LDMs with a Vision Transformer (*DiT*): patch the latent $\mathbf{z}$ into tokens, process with standard Transformer blocks, unpatch. Conditioning (class label or text) is injected via adaptive layer norm: $\gamma(\mathbf{c}), \beta(\mathbf{c})$ (scale and shift) are predicted by a small MLP of the condition embedding. DiT-XL/2 achieves FID $= 2.27$ on ImageNet 256×256 (class-conditional), surpassing the best U-Net LDM (3.60). The scaling behaviour of DiT follows the Chinchilla law: larger models and more compute reduce FID log-linearly. Stable Diffusion 3 (Esser et al., 2024) uses a *Multi-Modal DiT* (*MM-DiT*) that applies separate attention streams for text and image tokens, merged by cross-modal attention.

38.1 Neural Scaling Laws

Kaplan et al. (2020) and Hoffmann et al. (2022) discovered power-law relationships between compute, data, model size, and loss in language modelling.

The Kaplan scaling laws. For a transformer trained on C total compute (FLOPs), optimal loss scales as:

$$L(C) \approx \left(\frac{C_{\min}}{C}\right)^{\alpha_C}, \qquad \alpha_C \approx 0.050$$

For model parameters N:

$$L(N) \approx \left(\frac{N_c}{N}\right)^{\alpha_N}, \qquad \alpha_N \approx 0.076$$

For dataset size D (tokens):

$$L(D) \approx \left(\frac{D_c}{D}\right)^{\alpha_D}, \qquad \alpha_D \approx 0.095$$

These power laws hold over 6+ orders of magnitude in each variable. Key finding (Kaplan): for a fixed compute budget C, optimal allocation spends most on model size with relatively less on data.

Chinchilla laws (Hoffmann 2022). Re-examining Kaplan's setup with more careful evaluation, Hoffmann et al. found the optimal compute allocation is *equal* between model size and

tokens: $N_{\text{opt}} \propto C^{0.5}$, $D_{\text{opt}} \propto C^{0.5}$. The optimal tokens-per-parameter ratio is approximately 20 tokens per parameter. Chinchilla (70B, 1.4T tokens) outperforms Gopher (280B, 300B tokens) despite $4\times$ fewer parameters.

Beyond language. Scaling laws have been found for: code generation, vision (ViT), multimodal models, reinforcement learning from human feedback, protein language models, and chess. The power-law exponents vary by task and architecture but the structure (power law in N, D, C) is universal.

Emergent abilities. Wei et al. (2022) documented capabilities that appear suddenly at specific model scales ("emergence"): in-context few-shot learning at ~ 100B parameters, chain-of-thought reasoning at $\sim$50B parameters, instruction following at ~ 10B parameters. These are not smooth extrapolations of the loss curve — they appear as phase transitions in downstream task metrics.

Example — Compute-optimal training frontier

For training a model with total budget C FLOPs: allocate $N^* = C^{0.5} \times 10^{-10}$ parameters and $D^* = 20N^*$ tokens. For $C = 10^{23}$ FLOPs: $N^* \approx 70$B, $D^* \approx 1.4$T tokens — the Chinchilla optimal point. Over-training on smaller models (e.g. Llama 7B at 1T tokens) can yield better *inference* efficiency even at slightly higher training loss, since inference runs billions of times post-training.

Example — Emergent capabilities and phase transitions

Wei et al. (2022) documented that several abilities (multi-step arithmetic, chain-of-thought reasoning, BIG-Bench tasks) appear abruptly at specific scale thresholds rather than improving smoothly. For GPT-3–sized models ($\sim$ 175B parameters), 3-digit addition accuracy jumps from $< 5\%$ to $> 80\%$ between 10^{23} and 10^{24} training FLOPs. Schaeffer et al. (2023) argue that this is partly an artefact of using discontinuous metrics (exact match): with continuous metrics (token-level accuracy), the same transitions appear smooth. Nevertheless, the Chinchilla scaling law (§38.1) predicts performance continuously in compute budget; emergent capabilities may represent qualitative phase transitions in the loss landscape not captured by aggregate perplexity.

Compute-optimal training vs. inference-optimal training The Chinchilla law gives compute-optimal training at a given FLOP budget. But deployment flips the objective: *inference-optimal* models minimise cost per query. Since a smaller model runs faster, one can afford to overtrain it (more tokens per parameter) and still achieve the quality of a larger compute-optimal model, at lower inference cost. LLaMA (Touvron et al., 2023) explicitly overtrains 7B/13B/65B parameter models on 1.0T/1.4T/1.4T tokens (vs. Chinchilla optimal of 140B/260B/1.3T) — the 7B model achieves quality competitive with GPT-3 (175B) at $25\times$ lower inference FLOP. This *inference-compute Pareto frontier* is now the standard design philosophy for open-weight LLMs.

38.2 Hyperbolic Neural Networks

Hyperbolic space $\mathbb{H}^n_c$ (constant curvature $-c < 0$) has a fundamental property: the volume of a ball of radius r grows as $e^{r\sqrt{c}}$, exponentially faster than the polynomial growth r^n in Euclidean space. This makes hyperbolic space ideally suited for embedding *hierarchical* and *tree-structured* data with low distortion.

Poincaré ball model. The Poincaré ball $\mathbb{B}^n_c = \{\mathbf{x} \in \mathbb{R}^n : c\|\mathbf{x}\|^2 < 1\}$ has geodesic distance:

$$d_c(\mathbf{x}, \mathbf{y}) = \frac{2}{\sqrt{c}} \tanh^{-1}\left(\sqrt{c}\,\| -\mathbf{x} \oplus_c \mathbf{y}\|\right)$$

where the Möbius addition is $\mathbf{x} \oplus_c \mathbf{y} = \frac{(1+2c\langle \mathbf{x},\mathbf{y}\rangle + c\|\mathbf{y}\|^2)\mathbf{x} + (1-c\|\mathbf{x}\|^2)\mathbf{y}}{1+2c\langle \mathbf{x},\mathbf{y}\rangle + c^2\|\mathbf{x}\|^2\|\mathbf{y}\|^2}$.

Hyperbolic GNN (HGNN). Chami et al. (2019) lift the GCN message-passing equations to the Poincaré ball:

$$\mathbf{h}_i^{(l+1)} = \sigma^{\otimes_c}\left(\exp_{\mathbf{0}}^c\left(\mathbf{W}^{(l)} \log_{\mathbf{0}}^c(\mathbf{h}_i^{(l)}) + \mathbf{W}^{(l)}\, \tilde{\mathbf{m}}_i^{(l)}\right)\right)$$

The *logarithmic map* $\log_{\mathbf{0}}$ maps from $\mathbb{B}^n_c$ to the tangent space at the origin (a Euclidean space), where the linear transformation $\mathbf{W}$ is applied, and the *exponential map* $\exp_{\mathbf{0}}$ maps back. The aggregated message is:

$$\tilde{\mathbf{m}}_i^{(l)} = \log_{\mathbf{0}}^c\left(\bigoplus_{j \in \mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} \exp_{\mathbf{0}}^c(\mathbf{W}^{(l)} \log_{\mathbf{0}}^c(\mathbf{h}_j^{(l)}))\right)$$

Aggregation uses the *Fréchet mean* in hyperbolic space, approximated by the Einstein midpoint.

Embedding quality: Worst-case distortion

For a tree T with n nodes, any Euclidean embedding requires distortion $\Omega(\log n / \log\log n)$ (Linial et al., 1994). Sarkar (2011) showed that any tree embeds in $\mathbb{H}^2$ with distortion at most $(1+\varepsilon)$ using only 2 dimensions and $O(n\log(1/\varepsilon))$ precision bits. For WordNet (82k nouns, 671k hypernym pairs), Poincaré embeddings (Nickel & Kiela, 2017) achieve 87.6% reconstruction accuracy in $\mathbb{B}^2_1$ vs. 57.3% for 200-dimensional Euclidean word2vec.

Fully hyperbolic networks (FULLY-H). Liu et al. (2019) perform all operations — linear maps, attention, aggregation — natively in the Poincaré ball using Möbius transformations $f_{\mathbf{W},\mathbf{b}}(\mathbf{x}) = \mathbf{W} \otimes_c \mathbf{x} \oplus_c \mathbf{b}$, where $\mathbf{W} \otimes_c \mathbf{x} = \exp_{\mathbf{0}}^c(\mathbf{W}\log_{\mathbf{0}}^c(\mathbf{x}))$. This avoids repeated log/exp projections and achieves better numerical stability at high curvature.

Applications. Hyperbolic GNNs are competitive on: disease–gene association (GO ontology, strict hierarchy), drug-target interaction prediction (protein interaction DAG), few-shot image classification (imagenet class hierarchy), and conversational AI (intent taxonomy trees). The gain over Euclidean GNNs is most pronounced when the graph has high δ-hyperbolicity (trees have $\delta = 0$; small-world graphs have $\delta = O(\log n)$).

Lorentzian (hyperboloid) model vs Poincaré ball An alternative model of hyperbolic space is the *Lorentzian model* (hyperboloid): $\mathbb{H}^n = \{\mathbf{x} \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{x}\rangle_L = -1, x_0 > 0\}$ with Lorentzian inner product $\langle \mathbf{x}, \mathbf{y}\rangle_L = -x_0 y_0 + \sum_{i=1}^n x_i y_i$. Geodesic distance: $d(\mathbf{x},\mathbf{y}) = \operatorname{arccosh}(-\langle \mathbf{x}, \mathbf{y}\rangle_L)$. The hyperboloid model has better numerical properties than the Poincaré ball (no boundary singularity at $\|\mathbf{x}\| \to 1$) and exact gyrovector operations; the two models are related by stereographic projection. Law et al. (2019) show that the hyperboloid model has lower numerical error for large hierarchies (depth > 10).

38.3 Higher-Order GNNs and Graph Transformers

Standard MPNNs operate on individual nodes and 1-hop neighbourhoods — a limitation bounded by 1-WL. Higher-order GNNs break this barrier by operating on k-tuples of nodes, enabling detection of structural patterns (cycles, cliques) invisible to standard message passing.

k-dimensional GNN (k-GNN). Morris et al. (2019) define k-WL GNNs operating on k-tuples $\mathbf{s} \in \mathcal{V}^k$. For each tuple, the neighbourhood is:

$$\mathcal{N}_j(\mathbf{s}) = \{\mathbf{s}' \in \mathcal{V}^k : \mathbf{s}'_i = \mathbf{s}_i \ \forall i \neq j\}$$

(all k-tuples differing from $\mathbf{s}$ in exactly one position j). Message passing on this neighbourhood corresponds to the k-WL test. The k-GNN is at least as expressive as k-WL and strictly more expressive than $(k-1)$-WL. Complexity: $O(n^k d)$ per layer — exponential in k, practical only for $k = 2, 3$.

PPGN (Provably Powerful GNNs). Maron et al. (2019) characterise all equivariant linear layers on tensors $\mathbb{R}^{n^k}$ and show that with 2-dimensional tensor inputs and suitable nonlinearities, PPGN achieves 3-WL expressiveness in $O(n^3)$ per layer. PPGN can count triangles, 4-cycles, and other sub-graphs that 1-WL GNNs cannot.

Sub-graph GNNs (NGNN, DS-GNN)

Rather than k-tuples, sub-graph GNNs apply a standard MPNN independently to each *rooted sub-graph* $G[r]$ (the induced sub-graph on the r-hop neighbourhood of each node r). The node-level representation in $G[r]$ encodes both the local structure and the node's role within it. DS-GNN (Bevilacqua et al., 2022) shows this achieves 3-WL expressiveness for appropriate sub-graph policies, at cost $O(n \cdot |\mathcal{E}|d)$ — more scalable than full 3-GNN.

Graph Transformers as higher-order models. With Laplacian PE (§38.4), a Graph Transformer effectively operates on pairs of nodes (via the global attention matrix $\mathbf{A}_{ij}$), accessing 2-WL-equivalent structural information. With random-walk structural encodings, it can approximate 3-WL features. Empirically, GPS with RWPE achieves state-of-the-art on the LRGB benchmark (large-range graph benchmark), which specifically tests long-range reasoning beyond 1-WL.

Combinatorial complexity trade-off.

Model	WL power	Per-layer cost
GIN	1-WL	$O(\|\mathcal{E}\|d)$
2-GNN	2-WL	$O(n^2 d)$
3-GNN / PPGN	3-WL	$O(n^3 d)$
DS-GNN	3-WL (approx)	$O(n\|\mathcal{E}\|d)$
GPS + RWPE	$\approx$ 3-WL	$O((\|\mathcal{E}\| + n^2)d)$

Example — PPGN expressiveness on cycle counting

PPGN (Maron et al., 2019) achieves 3-WL expressiveness and can count cycles of any length. On the EXP dataset (60 pairs of non-isomorphic regular graphs indistinguishable by 1-WL), GIN achieves 50% accuracy (random guessing); PPGN achieves 100%. On ZINC-12K (molecular property regression), PPGN achieves MAE $= 0.083$ vs. GIN 0.387 and GCN 0.459, demonstrating that the ability to count rings (4-cycles, 6-cycles) is critical for molecular property prediction.

38.4 Position Encoding on Graphs: Laplacian Eigenvectors

Unlike sequences or grids, graphs have no natural coordinate system. Position encoding (PE) supplies graph Transformers with structural/positional context that pure message passing (bounded by 1-WL) cannot capture.

Laplacian eigenvector PE. Compute the k eigenvectors $\mathbf{U}_k = [\mathbf{u}_1, \dots, \mathbf{u}_k] \in \mathbb{R}^{n \times k}$ of the normalised graph Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, corresponding to the k smallest eigenvalues $0 = \lambda_1 \leq \cdots \leq \lambda_k$. Node v receives PE $\mathbf{p}_v = \mathbf{U}_k[v, :] \in \mathbb{R}^k$.

Why eigenvectors encode position.

- $\mathbf{u}_1 = \mathbf{1}/\sqrt{n}$ (constant — encodes no position).
- $\mathbf{u}_2$ (Fiedler vector) partitions the graph by its sign pattern, revealing the two most well-separated communities.
- Higher eigenvectors resolve finer structure. For the path graph P_n: $u_k(v) = \sqrt{2/n}\sin(k\pi v/n)$ — exactly the Fourier basis on a 1D grid, recovering sinusoidal PE as a special case.

Sign ambiguity. Eigenvectors are defined up to a sign flip $\mathbf{u}_k \mapsto -\mathbf{u}_k$. For degenerate eigenspaces (multiplicity > 1), the eigenvectors are defined up to arbitrary rotation within the eigenspace. This breaks equivariance: two isomorphic graphs may receive different PEs.

SignNet (Lim et al., 2022). Resolves sign ambiguity by learning a sign-invariant function of the eigenvectors using an equivariant deep set:

$$\text{SignNet}(\mathbf{U}_k) = \rho\left(\sum_{\mathbf{s}\in\{\pm 1\}^k} \phi(\mathbf{s} \odot \mathbf{U}_k)\right)$$

summing over all 2^k sign flips. For k small ($k \leq 16$), this is tractable. SignNet is provably more expressive than any fixed-sign PE.

Random Walk PE (RWPE)

The t-step random walk return probability for node v is $\text{RW}^t_{vv} = [(\mathbf{D}^{-1}\mathbf{A})^t]_{vv}$. The RWPE of node v is the vector $[\text{RW}^1_{vv}, \text{RW}^2_{vv}, \ldots, \text{RW}^T_{vv}]$. Unlike Laplacian PE, RWPE is already invariant to eigenvector sign and captures local structure at multiple scales (short walks ↔ local degree; longer walks ↔ global connectivity). GPS (Rampášek et al., 2022) uses RWPE with $T = 20$ as the default PE.

Comparison. Laplacian PE captures global spectral geometry; RWPE captures local structural roles; wL-based features (e.g. Colour Refinement) capture combinatorial isomorphism class. State-of-the-art graph Transformers typically combine RWPE + Laplacian PE + edge features, giving significantly richer input representations than plain adjacency information.

Example — Laplacian PE on the cycle graph C_n

For the cycle graph C_n: eigenvalues $\lambda_k = 2(1 - \cos(2\pi k/n))$, eigenvectors $u_k(v) = \frac{1}{\sqrt{n}} e^{i2\pi kv/n}$ (complex Fourier basis). Taking the real and imaginary parts as PE: $p_v = [\cos(2\pi v/n), \sin(2\pi v/n), \cos(4\pi v/n), \ldots]$ — exactly the sinusoidal positional encoding of the Transformer, applied to a discrete cycle. Laplacian PE generalises this to arbitrary graphs: the eigenvectors adapt to the graph's symmetry, providing a canonical positional coordinate system even for irregular topologies.

38.5 Summary Table: GNN Architectures and Expressiveness

The table below compares the main GNN architectures covered in Volume III across key dimensions.

Model	Aggregation	Equivariance	WL-power	Complexity/layer
GCN	Mean (normalised)	S_n	$<$ 1-WL	$O(\lvert\mathcal{E}\rvert d)$
GraphSAGE	Sample+Mean/Max	S_n	$<$ 1-WL	$O(S\lvert\mathcal{V}\rvert d)$
GAT	Attn-weighted mean	S_n	$<$ 1-WL	$O(\lvert\mathcal{E}\rvert d)$
GIN	Sum (injective)	S_n	$=$ 1-WL	$O(\lvert\mathcal{E}\rvert d)$
R-GCN	Per-rel. mean	S_n	$<$ 1-WL	$O(\lvert\mathcal{R}\rvert\lvert\mathcal{E}\rvert d)$
HAN	Meta-path attn	S_n	$<$ 1-WL (per path)	$O(P\lvert\mathcal{E}\rvert d)$
EGNN	SE(3)-equiv. msg	$SE(3) \times S_n$	$=$ 1-WL	$O(\lvert\mathcal{E}\rvert d)$
GPS (Graph Transf.)	MPNN+global attn	S_n	$>$ 1-WL (with PE)	$O(\lvert\mathcal{E}\rvert d + n^2 d)$
PPGN	Sub-graph (2-WL)	S_n	$=$ 2-WL	$O(n^3 d)$

Notation. S_n: permutation equivariance on node sets. $SE(3)$: equivariant to 3D rotations, reflections, translations. WL-power: expressive power relative to the k-WL test. Complexity: per-layer, excluding non-linearity, for a graph with $|\mathcal{V}|$ nodes, $|\mathcal{E}|$ edges, $|\mathcal{R}|$ relation types, P meta-paths, feature dimension d.

Practical considerations.

- For *node classification* on homogeneous graphs: GIN or GAT are strong baselines; GraphSAGE for inductive/large-scale.
- For *graph classification* (molecular property prediction): GIN or higher-order GNNs; EGNN for 3D molecular data.
- For *heterogeneous graphs* (knowledge graphs, recommendation): R-GCN or HAN.
- For *long-range dependencies*: GPS or Expander-augmented MPNN.
- For $SE(3)$*-sensitive tasks* (protein structure, force fields): EGNN, NequIP, or MACE.

38.6 Connections to Classical Computer Science

The expressiveness and limitations of GNNs connect deeply to classical combinatorics, logic, and complexity theory, revealing fundamental boundaries that cannot be overcome by architectural modifications alone.

GNNs and the WL hierarchy. The k-WL test corresponds to the Sherali-Adams hierarchy of LP relaxations for graph isomorphism. k-WL is equivalent to k-variable logic with counting quantifiers (C^k). No polynomial-time algorithm is known to distinguish all graphs (the graph isomorphism problem is in NP ∩ co-NP but not known to be in P), and 3-WL cannot distinguish all graphs in polynomial time on a Turing machine.

Decidability and circuit complexity. A GNN with k message-passing steps computes a function in the complexity class TC^0 (constant-depth threshold circuits) when the aggregation/update functions are computable in TC^0. This means GNNs *cannot* solve problems that require global coordination of information over more than k hops — e.g. detecting a Hamiltonian cycle in general requires $\Omega(n)$ rounds of message passing (Loukas, 2020).

Over-squashing. As the receptive field grows exponentially with depth, an exponential number of node representations are compressed into a fixed-size vector. Alon & Yahav (2021) formalise over-squashing: the Jacobian $\partial \mathbf{h}_v^{(k)} / \partial \mathbf{h}_u^{(0)}$ decays exponentially with the graph distance $d(u, v)$, proportional to $(Cd^{-1/2})^k$ where d is the minimum degree on the path. Solutions: rewiring (adding edges to reduce diameter), virtual nodes, or graph Transformers with global attention.

Graph Isomorphism and the Colour-Coding Technique

Alon et al. (1995) show that detecting paths of length k can be done in $O(2^k \cdot |\mathcal{E}|)$ time using colour-coding: randomly colour nodes with k colours, then use dynamic programming on colour-coding paths. This is relevant to higher-order GNNs: operating on k-tuples allows detection of k-paths but at exponential cost in k.

PAC learning on graphs. The VC dimension of GNNs on graph classification tasks scales as $\Theta(Ld^2 \log(1/\delta)/\varepsilon^2)$ where L is depth and d is feature width, giving PAC sample complexity $O(Ld^2/\varepsilon^2)$ — the same order as MLPs of comparable depth and width. However, GNN generalisation benefits additionally from the graph structure (smoothness of the target func-

tion w.r.t. the graph topology), captured by spectral bounds on the graph signal (Gama et al., 2020).

> MPNN and the LOCAL model of distributed computing In the LOCAL model of distributed computing, each node processes information only from its k-hop neighbourhood in k rounds. MPNNs with k layers are exactly k-round LOCAL algorithms. Classical lower bounds from distributed computing apply directly: deciding whether a graph has a perfect matching, computing a $(1+\varepsilon)$-approximation to the maximum matching, or 3-coloring a cycle all require $\Omega(\log^* n)$ or $\Omega(n)$ LOCAL rounds on adversarial graphs. These lower bounds prove that no bounded-depth GNN can solve these problems with bounded error on all graphs, regardless of the aggregation function — a theoretical limit that no architectural modification can circumvent.

38.7 Benchmark Datasets and Evaluation Protocol

Node classification benchmarks.

- *Cora, Citeseer, Pubmed*: citation graphs with bag-of-words node features and topic labels. Standard transductive splits (Kipf & Welling, 2017). Small ($n < 20$K), low-diameter, homophilic — easy for simple GCNs; not representative of modern challenges.
- *ogbn-arxiv* (OGB, Hu et al., 2020): 169K nodes, 1.2M edges, 40 subject-area classes. Temporal split (train $\leq$ 2017, val 2018, test $\geq$ 2019), preventing leakage. Degree distribution follows a power law; heterophily is moderate.
- *ogbn-proteins*: 132K proteins, 39M edges, 112 binary labels (disease associations). Edge features (8-dimensional, from STRING database); no node features. Tests models on dense, biology-derived graphs.
- *ogbn-papers100M*: 111M nodes, 1.6B edges — the largest public node-classification benchmark. Requires graph sampling (GraphSAGE-style) or cluster GNN approaches.

Graph classification benchmarks.

- *TU datasets* (MUTAG, PROTEINS, NCI1, IMDB-B): small molecules and social networks. Widely used but criticised for high variance due to small size; use 10-fold CV with careful splits (Errica et al., 2020).
- *ogbg-molhiv, ogbg-molpcba*: molecular property prediction (HIV inhibition, 128 assays). Scaffold split separates test molecules by chemical scaffold — a more rigorous OOD evaluation than random splits. Metric: ROC-AUC.
- *ZINC*: 250K molecules, regression of constrained solubility. Tests ability to count rings and functional groups; 12K training subset is standard.

Long-range graph benchmarks.

- *LRGB* (Dwivedi et al., 2022): five datasets designed to test long-range dependencies — Peptides-func (peptide function, 15K graphs, avg. diameter 57), Peptides-struct (3D structure regression), PASCALVOC-SP (superpixel graphs from VOC images, 11K graphs, ~ 450 nodes each), COCO-SP, and PCQM-Contact (molecular contacts, 529K pairs). Standard metrics: AP (multi-label), MAE, macro-F1.

Evaluation pitfalls.

- *Data leakage*: random graph splits allow test molecules with the same scaffold as training data; scaffold splits give more honest OOD performance.
- *Hyperparameter tuning*: many GNN papers tune on test set implicitly; OGB enforces a fixed validation-based selection.
- *Variance*: small TU datasets have $\pm 5\%$ standard deviation; reporting mean without confidence interval is uninformative. Run ≥ 10 seeds.
- *Baseline strength*: Errica et al. (2020) showed that a simple MLP (ignoring graph structure) matches or beats several published GNNs on TU benchmarks — a sobering finding motivating the move to OGB-style evaluation.

The OGB leaderboard and reproducibility The Open Graph Benchmark (Hu et al., 2020) enforces: (1) fixed dataset splits (no per-paper split optimisation); (2) model selection by validation performance only; (3) test performance reported as mean$\pm$std over 10 seeds; (4) compute budget reported in GPU-hours. As of 2024, the ogbn-arxiv leaderboard has > 200 submitted methods; top models achieve $> 77\%$ accuracy (vs. 72% for vanilla GCN). The leaderboard shows diminishing returns beyond $\sim 74\%$: gains come primarily from ensembling, label propagation, and LLM-enhanced node features — structural GNN innovations account for $< 1\%$ of the top-to-baseline gap.

38.8 Summary and Open Problems in Geometric Deep Learning

Volume III has developed the mathematics of graph and geometric learning from first principles, unifying spectral graph theory, group representation theory, optimal transport, Riemannian geometry, and differentiable rendering under the GDL blueprint.

Core results reviewed.

- Graph Laplacian: spectral decomposition, Cheeger inequality, graph signal processing.
- MPNN framework: unification of GCN, GAT, GraphSAGE, GIN; WL expressiveness bounds.
- Equivariant networks: G-CNN, Steerable CNNs, EGNN, SE(3)-Transformers, IPA.
- Optimal transport: Kantorovich duality, Sinkhorn, Gromov-Wasserstein, barycenters, sliced W_2.
- Riemannian geometry: geodesics, exponential map, Poincaré ball, Riemannian GD.
- Generative models: score matching, DDPM/DDIM, flow matching, latent diffusion.
- Multimodal learning: CLIP, Flamingo, BLIP-2, NeRF, SIREN.
- Scaling: efficient attention (Flash Attention, linear attention), GNN expressiveness beyond 1-WL.

Open problems.

Expressiveness. The gap between k-WL for $k \geq 3$ and practical polynomial-time architectures remains large. Can sub-graph GNNs (DS-GNN, NGNN) match 3-WL expressiveness

at $O(n|\mathcal{E}|)$ cost for all graph families? Is there a GNN architecture with super-polynomial expressiveness advantage over 1-WL without super-linear complexity?

Scalability of equivariant networks. SE(3)-equivariant networks (NequIP, MACE) achieve state-of-the-art on small molecules but scale poorly to proteins (> 500 residues) due to $O(n^2)$ all-pairs interactions. Efficient approximate equivariance (e.g. local frames + long-range attention) is an active research direction.

OT at scale. Gromov-Wasserstein for graphs with $n > 10^4$ nodes remains expensive. Scalable approximations (sliced GW, hierarchical GW) lose the theoretical guarantees of the original. Applications in single-cell genomics (aligning $> 10^5$ cells) require sub-quadratic methods.

Unification of generation paradigms. Diffusion, flow matching, consistency models, and masked autoregressive models each have different speed/quality trade-offs. A unified theoretical framework explaining when each is optimal — and how to convert between them losslessly — is lacking.

Generalisation theory for GNNs. PAC-Bayes bounds for GNNs depend on spectral properties of the graph, but existing bounds are often vacuous on large graphs. Tighter graph-structure-aware generalisation guarantees, especially for OOD (out-of-distribution) molecular graphs, remain an open problem.

Connections to Volumes IV and V. Volume IV (Optimisation and Robustness) connects directly: GNN training landscapes, adversarial robustness of graph models, and differential privacy for graph data. Volume V (Efficient and Emergent Intelligence) extends the transformer story: efficient attention mechanisms developed here (Flash Attention, linear attention) underpin the SSM/Mamba architectures and mechanistic interpretability of graph-processing circuits.

PART 39

Graph Generation and Neural Combinatorial Optimisation

We study generative models for graphs, neural approaches to combinatorial problems on graphs, and protein structure prediction beyond AlphaFold2.

The 3D generation frontier The most active open problem at the time of writing (2025) is *joint generation of protein sequences and structures*. FrameDiff (Yim et al., 2023) runs SE(3)-equivariant diffusion over backbone frames; FoldFlow (Bose et al., 2023) applies flow matching on $SE(3)^n$. Both produce novel protein backbones with high predicted designability (pLDDT $> 80\%$). The challenge: simultaneously generating amino-acid sequences that fold into these backbones, at the quality level of natural proteins. Bridging the gap between structure generation (geometric deep learning) and sequence generation (language modelling) is the central challenge connecting Volume III to Volume V.

39.1 Graph Generation: VGAE, GraphRNN, and DiGress

Generating graphs is fundamentally different from generating images or text: the output is a discrete combinatorial object (nodes and edges) without a canonical ordering, requiring special treatment of permutation invariance.

Variational Graph Autoencoder (VGAE) (Kipf & Welling, 2016). The simplest graph generative model encodes the graph into node embeddings and decodes edges via inner products:

$$q(\mathbf{Z}|\mathbf{X},\mathbf{A}) = \prod_i q(\mathbf{z}_i|\mathbf{X},\mathbf{A}), \quad q(\mathbf{z}_i|\mathbf{X},\mathbf{A}) = \mathcal{N}(\square_i, \mathrm{diag}(\square_i^2))$$

where $\mathbf{Z} = \mathrm{GCN}(\mathbf{X},\mathbf{A})$. The decoder predicts the adjacency: $\hat{A}_{ij} = \sigma(\mathbf{z}_i^\top \mathbf{z}_j)$. Training maximises the ELBO: $\mathbb{E}_q[\log p(\mathbf{A}|\mathbf{Z})] - \mathrm{KL}(q(\mathbf{Z}|\mathbf{X},\mathbf{A})\|p(\mathbf{Z}))$. VGAE is simple and effective for link prediction but produces dense graphs and cannot generate graphs of variable size.

GraphRNN (You et al., 2018). Generates graphs autoregressively: at each step t, add a node and decide which previous nodes to connect to it. Two RNNs are interleaved:

- *Graph-level RNN*: encodes the current graph state h_t^G.
- *Edge-level RNN*: conditioned on h_t^G, sequentially emits edge probabilities $p(A_{t,j} = 1)$ for $j = 1, \ldots, t-1$.

A BFS node ordering reduces the output sequence length from $O(n^2)$ to $O(n \cdot \text{BFS-width})$. GraphRNN generates chemically valid molecules and social network topologies with high fidelity, but struggles with long-range structural dependencies.

Diffusion-based graph generation. DiGress (Vignac et al., 2023) applies discrete diffusion to graphs: the forward process corrupts the adjacency and node/edge attributes by randomly adding or removing edges (categorical noise). The reverse process is a graph Transformer $p_\theta(G_{t-1}|G_t)$ trained with a cross-entropy loss. Key innovation: the *marginal* noise distribution at each step is set to the empirical distribution of the dataset (not uniform), so the model must learn only to denoise, not to reconstruct statistics from scratch.

Evaluation Metrics for Graph Generation

- *Validity*: fraction of generated graphs satisfying hard constraints (valency rules for molecules).
- *Uniqueness*: fraction of valid graphs not seen in the training set.
- *Novelty*: fraction of valid graphs not identical to training examples.
- *Fréchet Inception Distance* (*GraphFID*): distance between feature distributions of real and generated graphs, using a pre-trained GNN encoder.
- *NSPDK MMD*: maximum mean discrepancy on neighbourhood sub-graph pair kernel features — captures local topology statistics.

DiGress achieves $> 90\%$ validity on QM9 and MOSES benchmarks, outperforming all prior non-diffusion methods.

SE(3)-equivariant molecular generation. For 3D molecule generation, the model must produce atom types *and* 3D coordinates equivariantly. EDM (Hoogeboom et al., 2022) runs diffusion in the joint space $(x_1, \dots, x_n, h_1, \dots, h_n) \in (\mathbb{R}^3 \times \mathbb{R}^d)^n$, using an E(3)-equivariant score network (EGNN). The centre-of-mass is fixed at the origin (E(3)-invariant prior), ensuring the generated molecule has no preferred position in space.

Evaluation challenges for 3D molecule generation Evaluating 3D molecular generators requires checking: (1) *atom stability* — each atom has the correct valency; (2) *molecule stability* — all atoms are stable simultaneously; (3) *validity* — no steric clashes ($\min_{i \neq j} \|\mathbf{r}_i - \mathbf{r}_j\| > 0.4\,\text{Å}$); (4) *novelty vs. training set*; (5) *uniqueness*; (6) *diversity* (mean pairwise similarity < 0.5). EDM achieves 82% molecule stability on QM9 (9 heavy atoms). A key limitation: current metrics do not assess synthesisability — a molecule may be geometrically valid but chemically impossible to synthesise. SA score (synthetic accessibility) and SCScore are used as proxies.

39.2 Neural Combinatorial Optimisation on Graphs

Combinatorial optimisation problems on graphs — TSP, VRP, MIS, graph colouring — are NP-hard in general. Neural combinatorial optimisation (NCO) trains models end-to-end to produce near-optimal solutions, amortising the solve time over many instances.

Pointer Networks (Vinyals et al., 2015). For the TSP on n cities: encode all cities with a bidirectional LSTM, then decode a permutation autoregressively using *attention as a pointer*:

$$p(\pi_t | \pi_{1:t-1}, \mathbf{x}) = \text{softmax}(C \cdot \tanh(\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 \mathbf{d}_t))$$

where $\mathbf{h}_i$ are encoder states and $\mathbf{d}_t$ is the decoder state. Trained with REINFORCE ($\nabla_\theta \log p(\pi|\mathbf{x}) \cdot (R(\pi) - b(\mathbf{x}))$ with learned baseline b). Achieves near-optimal tours on $n \leq 50$ cities; degrades for larger instances.

Attention Model (POMO) (Kwon et al., 2020). Replaces the LSTM with a Transformer encoder/decoder and uses multiple starting points for rollout (POMO: Policy Optimisation with Multiple Optima). Augments each instance with 8 symmetry transformations (rotations, reflections) to reduce variance. Achieves $< 1\%$ optimality gap on TSP-100 instances, competitive with classical LKH3 solver in inference time.

GNN-guided Branch and Bound

Gasse et al. (2019) train a GNN to imitate the variable-selection policy of a strong branching oracle in MILP branch-and-bound. The bipartite variable-constraint graph is encoded by a GNN; the output is a ranking over branching candidates. This learned policy reduces the number of B&B nodes by 3–$10\times$ on MIPLIB instances, outperforming default CPLEX heuristics. The key insight: the variable-constraint interaction graph is a natural bipartite graph, exactly the domain where GNNs excel.

Diffusion for combinatorial optimisation. Sun & Yang (2023) train a diffusion model over binary edge masks to solve MIS (Maximum Independent Set). Starting from a random binary assignment, the reverse diffusion process iteratively refines the mask to satisfy independence constraints and maximise set size. On 3-regular graphs, this *difusco* approach achieves near-optimal MIS within 0.1% of the best known solution, outperforming greedy and GNN-based approaches.

Limitations. NCO approaches generalise poorly to instance sizes not seen during training — a Pointer Network trained on $n = 50$ degrades sharply at $n = 200$. Active research directions: (1) hierarchical decomposition (solve sub-problems, merge solutions); (2) curriculum training over increasing n; (3) hybrid solvers that use neural predictions to warm-start exact solvers.

Reinforcement learning for combinatorial optimisation NCO can be framed as a reinforcement learning problem: the state is the partial solution (e.g. partial tour in TSP), the action is the next element to add, the reward is the negative tour length at the end of the episode. Policy gradient methods (REINFORCE, PPO) train the policy network end-to-end. The main challenge is the *combinatorial reward signal*: the reward is only available at the end of the episode (sparse reward), making credit assignment difficult for long sequences. Curriculum training (start with small n, gradually increase) and reward shaping (intermediate rewards for sub-tour quality) are standard mitigations. LKH3 (Lin-Kernighan-Helsgott, the best exact solver) achieves $< 0.05\%$ optimality gap on TSP-1000 but requires > 10 CPU-seconds; NCO methods achieve $< 0.1\%$ gap in < 50ms on GPU.

39.3 Protein Structure Prediction: Beyond AlphaFold2

AlphaFold2 (§33) solved single-chain protein structure prediction. The frontier has shifted to: multimeric complexes, protein–ligand binding, protein design, and understanding protein dynamics.

AlphaFold-Multimer (Evans et al., 2022). Extends AF2 to complexes: chains are concatenated with a special inter-chain gap token; the pairwise representation $\mathbf{z}_{ij}$ encodes both intra- and inter-chain distances; template features are sourced from the PDB for complex structures. On the CASP15 multimer benchmark, AF-Multimer achieves interface TM-score > 0.8 on 85% of targets — a step change over prior docking methods.

RoseTTAFold2 (Baek et al., 2023). Generalises to protein–DNA, protein–RNA, and protein–small molecule complexes in a single model. The key extension: small-molecule atoms are included as additional tokens with SE(3)-equivariant coordinate refinement, using a modified IPA that handles both protein residue frames and ligand atom positions. RoseTTAFold2 achieves near-AF2 accuracy on single chains and near-AF-Multimer accuracy on complexes.

Protein Design with Inverse Folding

Given a backbone structure, *inverse folding* predicts the amino-acid sequence most likely to adopt that structure. ProteinMPNN (Dauparas et al., 2022) uses an SE(3)-invariant MPNN on the C_α–C_β backbone graph, achieving $> 50\%$ sequence recovery on held-out structures. By sampling from the sequence distribution of a designed backbone, ProteinMPNN enables de novo protein design: specify the desired backbone (e.g. a binding site), then recover a stable sequence. Validated experimentally with $> 70\%$ success rate on designed binding proteins.

Protein language models and structure. ESMFold (Lin et al., 2023) predicts 3D structure directly from a protein language model (ESM-2, 650M–15B parameters) without MSA search — $60\times$ faster than AF2. The structure module uses the same IPA architecture as AF2 but takes LM representations as input instead of Evoformer output. ESMFold accuracy is slightly below AF2 for well-studied protein families but competitive for orphan sequences with few homologs.

Conformational ensembles and dynamics. AF2 predicts a single lowest-energy conformation. Proteins are dynamic: enzymes sample multiple conformations along their catalytic cycle. MSA subsampling (Del Alamo et al., 2022) generates conformational diversity by deliberately reducing MSA depth, causing AF2 to hallucinate alternative structures. More principled approaches train flow matching or diffusion models over conformation space, conditioned on the sequence. EigenFold (Jing et al., 2023) uses a hierarchical diffusion over normal modes of the predicted structure, achieving ensemble generation competitive with MD simulation at a fraction of the computational cost.

PART 40

Multimodal Reasoning and Scientific Applications

We survey multimodal reasoning architectures and highlight scientific applications of geometric deep learning in chemistry, biology, and physics.

EXAMPLE — EigenFold vs. MD simulation on alanine dipeptide

Alanine dipeptide has two stable conformers ($C7_{eq}$ and $C7_{ax}$) separated by a ~ 5 kcal/mol barrier, sampled rarely in standard MD. EigenFold (Jing et al., 2023) generates an ensemble of 500 structures in < 10s; the ensemble correctly populates both basins with Boltzmann weights. Reference: 1ms ANTON MD simulation. EigenFold Boltzmann weight error: < 0.3 kcal/mol free-energy difference. By contrast, a 1μs Amber MD simulation (standard practice) undersamples $C7_{ax}$ by $8\times$ due to the rare-event barrier. The flow-matching approach treats conformation sampling as a transport problem from the AF2 single-structure prediction to the full Boltzmann distribution.

40.1 Multimodal Large Language Models: Architecture and Training

A **Multimodal Large Language Model (MLLM)** extends a text-only LLM with the ability to process images, video, audio, or documents as additional input modalities. The dominant paradigm (2023–2025) uses a frozen or lightly fine-tuned LLM backbone with a trainable visual adapter.

General architecture.

1. **Modality encoder**: a pre-trained vision encoder (e.g. ViT-L/14 from CLIP, SigLIP) extracts patch tokens $\mathbf{V} \in \mathbb{R}^{m \times d_V}$.
2. **Projection adapter**: maps $\mathbf{V}$ to the LLM's token space. Options: (a) linear projection (LLaVA 1.0); (b) MLP with 2 layers and GELU activation (LLaVA 1.5, InternVL); (c) cross-attention resampler (Flamingo, BLIP-2 Q-Former).
3. **LLM backbone**: receives the projected visual tokens prepended to the text tokens, processes them with causal self-attention.

Training stages. Most MLLMs use a two-stage recipe:

- *Stage 1* (*alignment pre-training*): train only the projection adapter on large image-caption datasets (LAION, CC3M/CC12M). Objective: next-token prediction on the caption given the image. Purpose: align the visual token distribution with the LLM's text token space.
- *Stage 2* (*instruction tuning*): fine-tune the adapter (and optionally the LLM) on instruction-following data (VQA, visual conversation, OCR, chart understanding). Datasets: LLaVA-Instruct (158K), ShareGPT4V (1.2M), CAULDRON (50 tasks).

LLaVA-1.5 Training Protocol

Liu et al. (2023) use: ViT-L/14@336 CLIP encoder (576 patch tokens per image); 2-layer MLP projection; Vicuna-13B LLM backbone. Stage 1: 558K image-caption pairs, adapter only, 10^{-3} lr, 1 epoch. Stage 2: 665K instruction pairs, adapter + LLM, 2×10^{-5} lr, 1 epoch. Total compute: $\sim 8\times$ A100 hours. LLaVA-1.5 achieves 85.9% on VQAv2

— competitive with models $10\times$ larger, demonstrating the efficiency of the two-stage recipe.

Dynamic resolution. Early MLLMs processed all images at a fixed 336×336 resolution, losing detail in high-resolution documents. *Dynamic high resolution* (InternVL, LLaVA-HD): tile large images into 2×2 or 3×3 crops, each processed independently by the ViT, then concatenate tile tokens with a downsampled global-view token. This scales to 1344×1344 effective resolution with $5\times$ more tokens.

Video and audio. For video, extract T frames uniformly; concatenate their patch tokens. For audio, compute a mel-spectrogram and treat it as an image (Whisper encoder) or train a dedicated audio ViT. The same LLM backbone handles all modalities when the adapter maps each to the same token space — a single model can reason about text, image, video, and audio jointly.

Token efficiency in MLLMs A 336×336 ViT/14 produces 576 visual tokens per image. For a 4K resolution document (3840×2160): with 6×6 tiling, this yields $36 \times 576 = 20736$ tokens — the LLM's context length is saturated before any text is added. *Token merging* (*ToMe*) (Bolya et al., 2023) progressively merges visually similar tokens using bipartite matching, reducing 576 tokens to ~ 150 with $< 0.5\%$ accuracy drop. *LLaVA-PruMerge* (Shang et al., 2024) selects tokens based on their attention to the CLS token in the ViT, retaining only the ~ 50 most salient. Token compression is the key engineering challenge for deploying MLLMs on high-resolution inputs.

40.2 Geometric Deep Learning in Scientific Discovery

GNNs and equivariant networks have achieved breakthroughs in scientific domains where the data has explicit geometric structure: molecules, crystals, quantum systems, and fluid dynamics.

Molecular property prediction. DimeNet++ (Gasteiger et al., 2020) incorporates angular (bond-angle) information by constructing a message-passing scheme over directed edges with angle embeddings $e_{ijk} = \text{RBF}(\|\mathbf{r}_{ij}\|) \otimes \text{SBF}(\theta_{ijk})$ (radial basis $\times$ spherical basis functions). This reaches chemical accuracy (< 1 kcal/mol MAE) on 11 of 12 QM9 targets. SphereNet adds dihedral angles; GemNet adds full 3D structural context.

Force field learning. NequIP (Batzner et al., 2022) uses $\text{SE}(3)$-equivariant convolutions with $l_{\max} = 3$ to predict per-atom energies and forces on MD trajectories. On the rMD17 benchmark (alanine dipeptide, aspirin, etc.), NequIP achieves force MAE < 5 meV/Å with 950 training configurations — $100\times$ fewer than SchNet, demonstrating that equivariance acts as a data-efficient inductive bias.

Crystal structure prediction. CGCNN (Xie & Grossman, 2018) encodes periodic crystal structures as graphs: atoms as nodes, bonds within a cutoff radius as edges, periodic boundary conditions handled by replicating unit cells. CGCNN predicts formation energy with MAE < 0.1 eV/atom on the Materials Project. GNoME (Merchant et al., 2023) extended this to discover 2.2 million new stable crystal structures, increasing the known stable inorganic crystal count by $\approx 45\times$.

Equivariant Neural Network for Hamiltonian Prediction

The electronic Hamiltonian $\mathbf{H}(\mathbf{R}) \in \mathbb{R}^{n_{\text{orb}} \times n_{\text{orb}}}$ of a molecule is a function of nuclear positions $\mathbf{R}$ that transforms as a rank-2 tensor under rotation. PhiSNet (Unke et al., 2021) uses an SE(3)-equivariant network to predict $\mathbf{H}$ directly, then solves the generalised eigenvalue problem $\mathbf{HC} = \mathbf{SCE}$ for orbital energies and coefficients. Accuracy: < 0.1 meV for frontier orbital energies on QM7-X, at $10^6\times$ the speed of DFT. The equivariance guarantee ensures that $\mathbf{H}(R\mathbf{R}) = \mathbf{D}(R)\mathbf{H}(\mathbf{R})\mathbf{D}(R)^\top$ for any rotation R, consistent with the physical transformation law.

Fluid dynamics and weather forecasting. GraphCast (Lam et al., 2023) models atmospheric dynamics on a hierarchical icosahedral mesh: encode ERA5 grid points to mesh nodes (GNN encoder), propagate on the mesh (12-layer GNN processor), decode back to grid (GNN decoder). Training on 39 years of ERA5 reanalysis at 0.25° resolution. GraphCast outperforms ECMWF's high-resolution deterministic forecast (HRES) at 1–10 day lead times for all 1380 forecast targets, at $1/1000$th the compute cost per forecast. The key: the icosahedral mesh provides quasi-uniform spatial resolution with long-range connectivity, allowing weather systems to propagate globally.

Quantum many-body systems. Neural quantum states (NQS) parameterise the many-body wave function $\psi_\theta(\mathbf{s}) = \langle \mathbf{s}|\psi\rangle$ as a neural network (originally RBM; now GNN or Transformer). For frustrated quantum magnets on lattices, PauliNet and FermiNet use equivariant architectures to enforce fermionic antisymmetry and spin symmetry, achieving ground-state energies competitive with quantum Monte Carlo on systems of > 100 electrons — previously inaccessible to variational methods.

Geometric deep learning and drug discovery pipelines A modern AI-assisted drug discovery pipeline integrates multiple geometric DL components: (1) *binding site prediction* (P2Rank, VolSite): GNN on protein surface point clouds to identify druggable pockets; (2) *docking* (DiffDock, FABind): SE(3)-diffusion model generating ligand poses inside the pocket; (3) *affinity prediction* (EGNN, SchNet): equivariant GNN predicting $\Delta G_{\text{binding}}$ from the docked complex; (4) *molecular optimisation* (REINVENT, Graph-GA): generative graph model iterating structures to improve predicted affinity and ADMET properties. End-to-end, this pipeline reduces the hit-to-lead phase from ~ 2 years to ~ 6 months, and has produced clinical candidates for COVID-19 main protease and KRAS inhibitors.

PART 41

Synthesis: Mathematical Themes and Unifying Perspectives

We draw together the mathematical threads of the volume and identify the structural analogies that connect seemingly disparate topics.

41.1 The Spectral-Spatial Duality in Graph Learning

Throughout Volume III, a recurring theme emerges: every operation defined spectrally has a spatial interpretation, and vice versa. Understanding this duality provides a unified perspective on GNN design.

Spectral convolution = polynomial spatial filter. A spectral graph convolution $h_\theta(\mathbf{L})$ with polynomial kernel $h(\lambda) = \sum_k \theta_k \lambda^k$ computes exactly a K-hop weighted average of node features: $[h(\mathbf{L})\mathbf{X}]_{vc} = \sum_{u:\, d(u,v)\leq K} \alpha_{vu} \mathbf{X}_{uc}$. The spectral filter coefficients θ_k determine the aggregation weights. GCN uses $h(\lambda) = 1 - \lambda/2$ (linear filter); ChebNet uses a degree-K polynomial. Spatial MPNNs are more general — they can use non-polynomial (learned) aggregation functions — but for linear aggregation, the two views are identical.

Eigendecomposition = disentanglement. The graph Fourier transform $\hat{\mathbf{f}} = \mathbf{U}^\top \mathbf{f}$ decomposes a node signal into frequency components. In the spatial view, $\hat{f}_k = \mathbf{u}_k^\top \mathbf{f}$ measures how much $\mathbf{f}$ "oscillates" at frequency λ_k — how dissimilar neighbouring nodes are in the $\mathbf{u}_k$ direction. Over-smoothing (§35.2) occurs when $h(\mathbf{L})$ suppresses high-frequency components aggressively ($h(\lambda) \to 0$ for $\lambda \to 2$), driving the signal toward the constant eigenspace $\mathbf{u}_1$.

Group Fourier transform = irrep decomposition. The graph Fourier transform generalises to groups: for a signal $f : G \to \mathbb{R}^d$, the group Fourier transform $\hat{f}_\rho = \sum_{g \in G} f(g)\rho(g)$ decomposes f into its irrep components. In an equivariant network, feature channels are grouped by irrep type (§28.4), and equivariant linear layers act independently within each type (Schur's lemma). The CG tensor product (§28.5) mixes irreps — the non-linear operation that allows information exchange across frequency channels, analogous to the non-linearity after a spectral convolution.

Optimal transport = spectral matching. The Wasserstein distance between two graph signals can be related to the spectral L^2 distance when the cost matrix is the graph distance: $W_1(\mu, \nu) = \|(\mathbf{L}^+)^{1/2}(\mu - \nu)\|_1$ (for the shortest-path cost), where $\mathbf{L}^+$ is the pseudo-inverse of the Laplacian. This *diffusion distance* formulation connects OT (§30) to spectral graph theory (§25–26): close distributions in W_1 differ mainly in high-frequency (fine-grained) components.

41.2 Equivariance as a Unifying Principle

The GDL blueprint (§35.1) frames all architectures as instances of equivariant signal processing. This section traces the equivariance principle across the volume's topics and identifies where it breaks down or must be relaxed.

Hierarchy of symmetry groups.

Architecture	Symmetry group $\mathfrak{G}$	Domain Ω
CNN	$\mathbb{Z}^2$ (translation)	$\mathbb{Z}^2$ grid
G-CNN (p4)	$p4$ (rot. + transl.)	$\mathbb{Z}^2$ grid
GNN (MPNN)	S_n (node permutation)	Graph G
EGNN	$SE(3) \times S_n$	3D point cloud
Spherical CNN	$SO(3)$	S^2
IPA (AF2)	$SE(3)^n$ (per-residue)	Protein backbone
ViT (Transformer)	S_n (token permutation)	Set of patches

Equivariance vs. invariance trade-off. Invariant functions (pooling layers, graph readouts) lose information — a scalar graph property cannot reconstruct the original graph. Equivariant functions (GNN layers, IPA) preserve information but may not converge as quickly (the output space is high-dimensional). Practical architectures cascade equivariant layers for local processing and invariant readouts for global prediction, exploiting both properties.

Where equivariance must be relaxed. Protein structure prediction requires $SE(3)$-equivariance globally, but the IPA module operates with *local frame* equivariance — each residue's output transforms under its own frame, not a global rotation. This is a *gauge symmetry*: the choice of local frame is arbitrary, and the physics is independent of this choice. Similarly, NeRF (§32.1–32.2) is not rotation-equivariant — the radiance field depends on the absolute direction of the viewing ray, not a relative direction. When the data genuinely lacks a symmetry, imposing it would harm expressiveness.

The emergence of positional encodings. For graphs and sequences without canonical ordering, the absence of a natural positional coordinate forces the introduction of learned or computed PEs (§38.4, §36.2). PEs break the full permutation symmetry S_n down to the trivial group, allowing the model to distinguish positions. The spectral Laplacian PE resolves this minimally: it is defined up to the symmetry of the graph's automorphism group, providing the finest positional information consistent with the graph's structure.

> Geometric deep learning and physics-informed networks Physics-Informed Neural Networks (PINNs) impose PDE constraints as soft losses (§32.3). The GDL blueprint provides a complementary approach: hard symmetry constraints via equivariant architectures. For Maxwell's equations on a 3D domain, an $SO(3) \times \mathbb{Z}_2$ equivariant network guarantees that the predicted electric and magnetic fields transform correctly under rotation and reflection — constraints that PINNs must enforce softly via auxiliary losses. Combining hard equivariance (architecture) with soft PDE constraints (loss) is an active research direction for scientific computing.

41.3 Mathematical Maturity and Prerequisites for Volume IV

Volume III has introduced a broad range of mathematical machinery. Before proceeding to Volume IV (Optimisation Theory and Robustness), it is worth consolidating the key tools that will reappear in different guises.

From spectral graph theory (§25–26). The Laplacian eigendecomposition underpins the graph signal processing framework and spectral GNNs. In Volume IV, the *graph stability* of GNNs — their robustness to graph perturbations (edge additions, node feature noise) — is analysed via perturbation theory of the Laplacian spectrum. The spectral gap λ_2 (Fiedler value) controls convergence of graph-based optimisation algorithms (gossip algorithms, distributed SGD), connecting to Volume IV's chapter on optimisation on graphs.

From optimal transport (§30). The Sinkhorn algorithm is a proximal point method on the KL divergence — a connection that recurs in Volume IV's treatment of mirror descent and Bregman divergences. The Wasserstein distance as a loss function (for generative models, domain adaptation) inherits the smoothness properties of OT: it is differentiable almost everywhere and its gradient is the optimal transport plan $\mathbf{P}^*$, making it directly usable in first-order optimisation. Distributional robustness (Volume IV §X) uses Wasserstein balls as uncertainty sets.

From equivariant networks (§28–29, §33). The Clebsch-Gordan decomposition (§28.5) is the key operation for mixing feature channels of different equivariance types. In Volume IV, this connects to *symmetry-aware adversarial examples*: perturbations that are small in the equivariant norm but large in the Euclidean norm can fool equivariant networks that rely on equivariance as an inductive bias. Certified robustness for equivariant networks uses the same representation-theoretic tools.

From Riemannian geometry (§34). Riemannian gradient descent and the Poincaré ball model recur in Volume IV's treatment of *natural gradient* methods: the natural gradient $\tilde{\nabla}_\theta \mathcal{L} = \mathbf{F}^{-1}\nabla_\theta \mathcal{L}$ (where $\mathbf{F}$ is the Fisher information matrix) is the Riemannian gradient of $\mathcal{L}$ on the statistical manifold equipped with the Fisher-Rao metric. K-FAC and EKFAC approximate $\mathbf{F}^{-1}$ efficiently, making natural gradient methods practical for deep networks.

From generative models (§37). The score function $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ is the central object in both diffusion models and Stein variational gradient descent (SVGD). Volume IV studies score-based methods for sampling from unnormalised distributions (e.g. Bayesian posteriors), using the Langevin dynamics $d\mathbf{x}_t = \nabla_{\mathbf{x}} \log p(\mathbf{x}_t)\, dt + \sqrt{2}\, d\mathbf{B}_t$ — the continuous-time limit of denoising diffusion.

Key inequalities reviewed

The following inequalities from Volume III recur throughout the series:

- *Cheeger inequality*: $\lambda_2/2 \le h(G) \le \sqrt{2\lambda_2}$ — spectral gap bounds expansion.
- *Poincaré inequality on graphs*: $\|\mathbf{f} - \bar{f}\mathbf{1}\|^2 \le \frac{1}{\lambda_2}\mathbf{f}^\top \mathbf{L}\mathbf{f}$ — controls convergence of graph-diffusion smoothing.
- *Stability theorem* (OT): $d_\infty(\mathrm{dgm}(\mathcal{X}), \mathrm{dgm}(\mathcal{Y})) \le \|\mathcal{X} - \mathcal{Y}\|_\infty$ — TDA descriptors are Lipschitz in input.
- *Sinkhorn convergence*: error $\le 2n^2 \exp(-k\varepsilon/(4n^2))$ after k iterations.
- *Riemannian GD rate*: $\frac{1}{T}\sum_t \|\mathrm{grad}\, f\|^2 \le \frac{2(f_0 - f^*)}{\alpha T} + \frac{\alpha L^2}{2}$ — same $O(1/\sqrt{T})$ as Euclidean GD.

Appendici Tecniche

Dimostrazioni Differite

A.1 Disuguaglianza di Cheeger per Grafi

Enunciato formale

Sia G un grafo non orientato connesso, $\mathbf{L}$ il suo Laplaciano normalizzato con secondo autovalore λ_2, e $h(G)$ la costante di Cheeger (espansività). Allora:

$$\frac{\lambda_2}{2} \;\leq\; h(G) \;\leq\; \sqrt{2\lambda_2}.$$

Dimostrazione del bound inferiore. Sia $S \subseteq V$ con $|S| \leq |V|/2$. Per la variazione di Rayleigh:

$$\lambda_2 = \min_{\mathbf{f} \perp \mathbf{1}} \frac{\mathbf{f}^\top \mathbf{L} \mathbf{f}}{\mathbf{f}^\top \mathbf{f}}.$$

Poniamo $f_v = 1/|S|$ se $v \in S$, $f_v = -1/|\bar{S}|$ altrimenti, e $\mathbf{f}$ normalizzato ad avere media zero. Un calcolo diretto mostra che il quoziente di Rayleigh di $\mathbf{f}$ è $\leq 2h(G)$, dunque $\lambda_2 \leq 2h(G)$.

Bound superiore (Cheeger discreta). Il bound superiore $h(G) \leq \sqrt{2\lambda_2}$ segue dalla costruzione di Cheeger: dato l'autovettore $\mathbf{u}_2$ associato a λ_2, ordinare i nodi per valore crescente di $u_{2,v}$ e scegliere il taglio di soglia t che minimizza il rapporto taglio/volume. L'analisi del valor medio di questo taglio su tutti i t produce la stima $h(G) \leq \sqrt{2\lambda_2}$. $\square$

A.2 Convergenza dell'algoritmo di Sinkhorn

Sia $\mathbf{P}^{(k)}$ la matrice di trasporto dopo k iterazioni di Sinkhorn con regolarizzazione entropica ε. Allora:

$$\|\mathbf{P}^{(k)} - \mathbf{P}^*\|_F \;\leq\; 2n^2 \exp\!\left(-\frac{k\varepsilon}{4n^2}\right),$$

dove $\mathbf{P}^*$ è la soluzione ottima e n la dimensione dei vettori di probabilità.

Dimostrazione (schema). L'algoritmo di Sinkhorn corrisponde alla proiezione alternata di Bregman sul prodotto di simplessi $\mathcal{U}(\mathbf{a})$ e $\mathcal{V}(\mathbf{b})$, rispetto alla divergenza KL. Per le proiezioni di Bregman, l'errore decresce geometricamente con tasso $\exp(-\varepsilon/(4n^2))$. Il fattore $2n^2$ al numeratore è una costante dipendente dalla dimensione del problema. $\square$

In pratica, $k = 100$–500 iterazioni con $\varepsilon = 0.01$ sono sufficienti per problemi di taglia $n \leq 1000$. Per n molto grandi (trasporto su nuvole di punti 3D) si usano varianti stocastiche o multi-scala.

A.3 Lemma di Schur e decomposizione in irriducibili

Lemma di Schur

Siano $\rho_1 : G \to GL(V_1)$ e $\rho_2 : G \to GL(V_2)$ due rappresentazioni irriducibili di un gruppo finito G. Sia $\phi : V_1 \to V_2$ un omomorfismo equivariante, i.e. $\phi(\rho_1(g)\mathbf{v}) = \rho_2(g)\phi(\mathbf{v})$ per ogni $g \in G$ e $\mathbf{v} \in V_1$. Allora:

1. Se $\rho_1 \not\cong \rho_2$, allora $\phi = 0$.
2. Se $\rho_1 \cong \rho_2$, allora $\phi = c\,\mathrm{Id}$ per qualche $c \in \mathbb{C}$.

Conseguenza per le reti equivarianti. Uno strato lineare equivariante $W : V \to W$ tra spazi di feature che si trasformano secondo $\rho_V = \bigoplus_i \rho_i^{m_i}$ e $\rho_W = \bigoplus_j \rho_j^{n_j}$ ha la forma a blocchi diagonali: $W = \bigoplus_\rho \mathbf{M}_\rho \otimes \mathbf{I}_{d_\rho}$, dove $\mathbf{M}_\rho \in \mathbb{R}^{n_\rho \times m_\rho}$ è la matrice di molteplicità e d_ρ

è la dimensione dell'irriducibile ρ. Il numero di parametri liberi è $\sum_\rho n_\rho m_\rho$, drasticamente inferiore alla matrice piena $(\sum_\rho n_\rho d_\rho) \times (\sum_\rho m_\rho d_\rho)$. □

A.4 Teorema di stabilità della persistent homology

Siano $\mathcal{X}$ e $\mathcal{Y}$ due spazi metrici compatti. I rispettivi diagrammi di persistenza $\mathrm{dgm}(\mathcal{X})$ e $\mathrm{dgm}(\mathcal{Y})$ soddisfano:

$$d_b(\mathrm{dgm}(\mathcal{X}), \mathrm{dgm}(\mathcal{Y})) \;\leq\; d_{GH}(\mathcal{X}, \mathcal{Y}),$$

dove d_b è la distanza bottleneck e d_{GH} è la distanza di Gromov-Hausdorff.

Corollario per TDA. Se si perturba un insieme di punti $\mathcal{X}$ aggiungendo rumore boundato: $\|\mathcal{X}' - \mathcal{X}\|_\infty \leq \delta$, allora $d_b(\mathrm{dgm}(\mathcal{X}'), \mathrm{dgm}(\mathcal{X})) \leq \delta$. Pertanto le feature topologiche (componenti connesse, loops, cavità) estratte dalla persistent homology sono δ-stabili al rumore. □

Ricettario: Operazioni su Grafi e Geometria

B.1 Costruzione del Laplaciano

Dato un grafo $G = (V, E)$ con matrice di adiacenza $\mathbf{A}$ e matrice dei gradi $\mathbf{D} = \mathrm{diag}(d_1, \dots, d_n)$:

Variante	Formula	Proprietà
Laplaciano combinatoriale	$\mathbf{L} = \mathbf{D} - \mathbf{A}$	PSD, $\mathbf{1}$ nel nucleo
Laplaciano normalizzato simm.	$\tilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$	Autovalori in $[0, 2]$
Laplaciano random walk	$\mathbf{L}_{\mathrm{rw}} = \mathbf{D}^{-1}\mathbf{L}$	Non simmetrico
Laplaciano con self-loop	$\hat{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$	$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$

B.2 Propagazione MPNN generalizzata

Lo schema generale di message passing al passo t per il nodo v:

$$\begin{aligned}
\mathbf{m}_{uv}^{(t)} &= \phi_m\Big(\mathbf{h}_v^{(t)}, \mathbf{h}_u^{(t)}, \mathbf{e}_{uv}\Big) \qquad u \in \mathcal{N}(v) \\
\mathbf{m}_v^{(t)} &= \bigoplus_{u \in \mathcal{N}(v)} \mathbf{m}_{uv}^{(t)} \\
\mathbf{h}_v^{(t+1)} &= \phi_u\Big(\mathbf{h}_v^{(t)}, \mathbf{m}_v^{(t)}\Big)
\end{aligned}$$

dove $\bigoplus$ è un'operazione di aggregazione (somma, media, max, attention). Istanze notevoli:

Modello	ϕ_m	Aggregazione $\bigoplus$
GCN	$\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}\mathbf{W}\mathbf{h}$	Somma pesata (fissa)
GAT	$\alpha_{uv}\mathbf{W}\mathbf{h}_u$	Somma pesata (learned)
GIN	$\mathbf{W}((1+\epsilon)\mathbf{h}_v + \sum \mathbf{h}_u)$	Somma
GraphSAGE	$[\mathbf{h}_v \Vert \mathrm{MEAN}(\mathbf{h}_u)]$	Media + concat
EGNN	$\phi_e(\mathbf{h}_v, \mathbf{h}_u, \Vert\mathbf{x}_v - \mathbf{x}_u\Vert^2)$	Somma (inv.)

B.3 Prodotto tensoriale di Clebsch-Gordan

Per due feature equivarianti di tipo l_1 e l_2 (ordine degli armonici sferici), il prodotto tensoriale si decompone in irriducibili:

$$V^{l_1} \otimes V^{l_2} = \bigoplus_{l=|l_1-l_2|}^{l_1+l_2} C^{l_1 l_2 l} \cdot V^{l},$$

dove $C^{l_1 l_2 l} \in \mathbb{R}^{(2l_1+1)(2l_2+1)\times(2l+1)}$ sono i coefficienti di Clebsch-Gordan. Nelle implementazioni (e3nn, MACE), questa operazione è pre-calcolata e applicata in batch.

B.4 Distanza di Wasserstein: formulazioni equivalenti

Formulazione	Espressione
Problema primale	$W_p(\mu,\nu) = \left(\inf_{\gamma\in\Pi(\mu,\nu)} \int \|x-y\|^p \, d\gamma\right)^{1/p}$
Duale (Kantorovich-Rubinstein, $p=1$)	$W_1 = \sup_{\|f\|_L \le 1} \mathbb{E}_\mu[f] - \mathbb{E}_\nu[f]$
Regolarizzato (Sinkhorn)	$W_\varepsilon = \min_{\gamma\in\Pi} \langle \mathbf{C}, \gamma\rangle - \varepsilon H(\gamma)$
Sliced ($p=2$)	$SW_2^2 = \int_{S^{d-1}} W_2^2(P_\theta \sharp \mu, P_\theta \sharp \nu)\, d\theta$
Gromov-Wasserstein	$GW = \min_\gamma \sum_{i,j,k,l} (c_1(x_i,x_k) - c_2(y_j,y_l))^2 \gamma_{ij}\gamma_{kl}$

Complessità: GNN, Attenzione e Trasporto Ottimale

Operazione	Tempo	Memoria	Note
Graph Neural Networks			
GCN (singolo strato)	$O(\|E\|d + nd^2)$	$O(nd)$	Sparse matmul
GAT (h teste)	$O(h\,\|E\|\,d)$	$O(\|E\|h + nd)$	Score per arco
GIN	$O(\|E\|d + nd^2)$	$O(nd)$	Come GCN
Spettrale completa	$O(n^2 d)$	$O(n^2)$	Evitare per $n > 10^4$
ChebNet (K-hop)	$O(K\|E\|d)$	$O(nd)$	K matmul sparse
Attenzione su grafi e sequenze			
Transformer std.	$O(T^2 d)$	$O(T^2)$	T = lunghezza
Sparse attention	$O(T\sqrt{T}\,d)$	$O(T\sqrt{T})$	Bigbird, Longformer
Linear attention	$O(Td^2)$	$O(d^2)$	Performer, FNet
Graph Transformer	$O((n+\|E\|)d)$	$O(nd)$	Attenzione locale
Trasporto ottimale			
OT esatto (LP)	$O(n^3 \log n)$	$O(n^2)$	Intractable $n > 10^3$
Sinkhorn (k iter.)	$O(kn^2)$	$O(n^2)$	$k \approx 100$ nella pratica
Sliced Wasserstein	$O(Ln \log n)$	$O(Ln)$	L proiezioni
Gromov-Wasserstein	$O(kn^3)$	$O(n^2)$	Per iter. Sinkhorn
Equivarianza e geometria			
CG tensor product	$O(d_{\text{in}}^2 d_{\text{out}} L^3)$	$O(d^2 L^2)$	L = ordine max
EGNN (propagazione)	$O(\|E\|d)$	$O(nd)$	Inv. su distanze
IPA (AF2, N_r res.)	$O(N_r^2 d)$	$O(N_r^2)$	Come Transformer
Sinkhorn su S^2	$O(kn_\Omega^2)$	$O(n_\Omega^2)$	n_Ω = campioni sfera

C.1 Scalabilità per grafi di grandi dimensioni

Per grafi con $n > 10^5$ nodi (reti sociali, knowledge graph), le tecniche di scalabilità più usate sono:

Mini-batch sampling (GraphSAGE): campionare k vicini per ogni nodo a ogni strato. Complessità per step: $O(k^L \cdot d^2)$ invece di $O(|E|\, d)$.

Cluster-GCN: partizionare il grafo in cluster con METIS; eseguire GCN all'interno di ciascun cluster. Riduce la lunghezza delle comunicazioni tra cluster.

GraphFM / SIGN: pre-calcolare le aggregazioni multi-hop $\mathbf{A}^k\mathbf{X}$ offline, poi applicare un MLP per nodo senza propagazione durante il training.

Glossario dei Simboli — Volume III

Simbolo	Significato
$G = (V, E)$	Grafo con nodi V e archi E
$n = \lvert V\rvert,\ m = \lvert E\rvert$	Numero di nodi e archi
$\mathbf{A} \in \{0,1\}^{n\times n}$	Matrice di adiacenza
$\mathbf{D}$	Matrice diagonale dei gradi
$\mathbf{L}$	Laplaciano del grafo
$\lambda_1 \le \cdots \le \lambda_n$	Autovalori del Laplaciano
$\mathbf{U}$	Matrice degli autovettori del Laplaciano
$\hat{\mathbf{f}} = \mathbf{U}^\top \mathbf{f}$	Trasformata di Fourier su grafo
$h(G)$	Costante di Cheeger (espansività)
$\mathcal{N}(v)$	Insieme dei vicini del nodo v
$\mathbf{h}_v^{(l)}$	Feature del nodo v allo strato l
$\mathbf{e}_{uv}$	Feature dell'arco (u, v)
$\bigoplus$	Operazione di aggregazione (MPNN)
G	Gruppo (teoria dei gruppi)
$\rho : G \to GL(V)$	Rappresentazione di gruppo
ρ_l	Rappresentazione irriducibile di ordine l
$C^{l_1 l_2 l}$	Coefficienti di Clebsch-Gordan
$Y_l^m(\hat{\mathbf{r}})$	Armonico sferico di ordine l, grado m
$D^l(R)$	Matrice di Wigner per $R \in SO(3)$
$SE(3)$	Gruppo di isometrie in $\mathbb{R}^3$
μ, ν	Misure di probabilità (OT)
$\Pi(\mu, \nu)$	Insieme dei piani di trasporto
$W_p(\mu, \nu)$	Distanza di Wasserstein-p
$GW(\mu, \nu)$	Distanza di Gromov-Wasserstein
$\mathbf{C} \in \mathbb{R}^{n\times m}$	Matrice dei costi (OT)
$\mathbf{P}^*$	Piano di trasporto ottimo
$H(\gamma)$	Entropia della misura γ
ε	Parametro di regolarizzazione entropica
$\mathcal{M}$	Varietà differenziabile
$T_p\mathcal{M}$	Spazio tangente in p
$\exp_p(\mathbf{v})$	Mappa esponenziale
$\log_p(q)$	Mappa logaritmica
$d_{\mathcal{M}(p,q)}$	Distanza geodetica

κ	Curvatura sezionale
$\mathbb{H}^d_c$	Spazio iperbolico di dimensione d, curvatura c
$\mathrm{dgm}(\mathcal{X})$	Diagramma di persistenza
$d_b(\cdot,\cdot)$	Distanza bottleneck
$d_{GH}(\cdot,\cdot)$	Distanza di Gromov-Hausdorff
β_k	Numero di Betti di ordine k
$\mathbf{V} \in \mathbb{R}^{m \times d_V}$	Token visivi (MLLM)
T	Lunghezza sequenza / numero frame
h	Numero di teste (multi-head attention)

Bibliografia

Teoria spettrale dei grafi e Laplaciano

1. F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.
2. U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.
3. M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

Graph Neural Networks: fondamenti

4. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2017.
5. J. Gilmer, S. S. Schütt, F. Brockherde, A. Tkatchenko, K.-R. Müller, and D. Maclaurin, "Neural message passing for quantum chemistry," in *Proc. ICML*, pp. 1263–1272, 2017.
6. M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NeurIPS*, 2016.
7. K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. ICLR*, 2019.
8. W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NeurIPS*, pp. 1024–1034, 2017.

Graph Attention Networks

9. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018.
10. S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" in *Proc. ICLR*, 2022.
11. Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun, "Masked label prediction: Unified message passing model for semi-supervised classification," in *Proc. IJCAI*, 2021.

Deep Learning geometrico ed equivariante

12. T. S. Cohen and M. Welling, "Group equivariant convolutional networks," in *Proc. ICML*, pp. 2990–2999, 2016.

13. M. M. Bronstein, J. Bruna, T. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

14. M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. arXiv:2104.13478, 2021.

15. V. G. Satorras, E. Hoogeboom, and M. Welling, "E(n) equivariant graph neural networks," in *Proc. ICML*, pp. 9323–9332, 2021.

16. N. Thomas *et al.*, "Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds," *arXiv preprint arXiv:1802.08219*, 2018.

Espressività e test di Weisfeiler-Lehman

17. B. Weisfeiler and A. Lehman, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsia*, vol. 2, no. 9, pp. 12–16, 1968.

18. C. Morris *et al.*, "Weisfeiler and Leman go neural: Higher-order graph neural networks," in *Proc. AAAI*, pp. 4602–4609, 2019.

19. H. Maron *et al.*, "Provably powerful graph networks," in *Proc. NeurIPS*, 2019.

20. D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," in *Proc. NeurIPS*, 2021.

Trasporto ottimale

21. C. Villani, *Optimal Transport: Old and New*. Springer, 2009.

22. M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *Proc. NeurIPS*, 2013.

23. G. Peyré and M. Cuturi, "Computational optimal transport," *Foundations and Trends in Machine Learning*, vol. 11, nos. 5–6, pp. 355–607, 2019.

24. F. Mémoli, "Gromov-Wasserstein distances and the metric approach to object matching," *Foundations of Computational Mathematics*, vol. 11, no. 4, pp. 417–487, 2011.

25. N. Bonneel, J. Rabin, G. Peyré, and H. Pfister, "Sliced and Radon Wasserstein barycenters of measures," *J. Mathematical Imaging and Vision*, vol. 51, no. 1, pp. 22–45, 2015.

Topological Data Analysis

26. H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction*. American Mathematical Society, 2010.

27. F. Chazal and B. Michel, "An introduction to topological data analysis: Fundamental and practical aspects for data scientists," *Frontiers in Artificial Intelligence*, vol. 4, 2021.

28. C. Chen and M. Kerber, "Persistent homology computation with a twist," in *Proc. European Workshop on Computational Geometry*, 2011.

Apprendimento multimodale e modelli vision-language

29. A. Radford *et al.*, "Learning transferable visual models from natural language supervision (CLIP)," in *Proc. ICML*, pp. 8748–8763, 2021.

30. J.-B. Alayrac *et al.*, "Flamingo: A visual language model for few-shot learning," in *Proc. NeurIPS*, 2022.

31. H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning (LLaVA)," in *Proc. NeurIPS*, 2023.

32. D. Bolya *et al.*, "Token merging: Your ViT but faster," in *Proc. ICLR*, 2023.

Rappresentazioni implicite neurali

33. B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. ECCV*, pp. 405–421, 2020.

34. V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions (SIREN)," in *Proc. NeurIPS*, 2020.

35. M. Tancik *et al.*, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Proc. NeurIPS*, 2020.

Previsione della struttura proteica

36. J. Jumper *et al.*, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, pp. 583–589, 2021.

37. M. Baek *et al.*, "Accurate prediction of protein structures and interactions using a three-track neural network (RoseTTAFold)," *Science*, vol. 373, no. 6557, pp. 871–876, 2021.

38. B. Jing, E. Erives, P. Pao-Huang, G. Corso, B. Berger, and T. Jaakkola, "EigenFold: Generative protein structure prediction with diffusion models," *arXiv preprint arXiv:2304.02198*, 2023.

Applicazioni scientifiche del Geometric DL

39. S. Batzner *et al.*, "E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials (NequIP)," *Nature Communications*, vol. 13, no. 2453, 2022.

40. J. Gasteiger, S. Groß, and S. Günnemann, "Directional message passing for molecular graphs (DimeNet)," in *Proc. ICLR*, 2020.

41. R. Lam *et al.*, "Learning skillful medium-range global weather forecasting (GraphCast)," *Science*, vol. 382, pp. 1416–1421, 2023.

42. A. Merchant *et al.*, "Scaling deep learning for materials discovery (GNoME)," *Nature*, vol. 624, pp. 80–85, 2023.

Geometria iperbolica e scaling

43. O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," in *Proc. NeurIPS*, pp. 5345–5355, 2018.

44. J. Kaplan *et al.*, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

45. G. Peng *et al.*, "Hyperbolic graph neural networks," in *Proc. NeurIPS*, 2019.

Testi di riferimento

46. M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. arXiv:2104.13478, 2021.

47. C. Villani, *Topics in Optimal Transportation*. American Mathematical Society, 2003.

48. J. M. Lee, *Introduction to Smooth Manifolds*, 2nd ed. Springer, 2013.

49. G. Peyré and M. Cuturi, *Computational Optimal Transport*. Now Publishers, 2019.

50. P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

Mathematics of Neural Networks — Volume III of V
Silvano Salvador